STATA 9 for surveys manual

Part 2

By

Sandro Leidi[1], Roger Stern[1],
Brigid McDermott[2],
Savitri Abeyasekera[1]

[1]Statistical Services Centre, University of Reading, U.K.

[2]Biometrics Unit Consultancy Services, University of Nairobi, Kenya

May 2006

ISBN 0-7049-9838-6

Contents

Preface

This is Part 2 of the ***Stata for Surveys*** guide and includes Chapters 11 to 20. The overall guide, comprising this manual together with Part 1 of the guide, is designed to support the use of Stata for the analysis of survey data. We envisage two sorts of reader. Some may already be committed to using Stata, while others may be evaluating Stata, in comparison to other software.

The original impetus for this guide was from the Central Bureau of Statistics (CBS) in Kenya. In an internal review in July 2002, they recommended that Stata be considered as one of the statistics packages they could use for their data processing. The case for Stata was based on Version 7, which was the current version when their review was undertaken. This case was strengthened by the introduction of Version 8, where the inclusion of menus, and the revision of the graphics were both particularly relevant. It was therefore agreed that Stata be introduced to their staff on training courses in 2004. These courses were planned jointly by them, together with the Statistical Services Centre (SSC), Reading, UK, and the Biometry Unit Consultancy Services (BUCS) at the University of Nairobi in Kenya.

The initial plan was to prepare notes and practical work for a 3-day course on Stata. This was to be followed by a 2-week course on data analysis using Stata. The idea to make the notes into a book came from Hills and Stavola (2004). The latest version of their book is called "A Short Introduction to Stata 8 for biostatistics". We found the organisation of the materials to be exactly what we needed for teaching surveys. We therefore suggested that we would try to have the same structure for this book, and that this consistency in approach might indeed help readers who might wish to use materials from the two books. We are most grateful to the authors and publishers of Hills and Stavola (2003), for agreeing to our request, and for sending a preprint of the Version 8 book, so we could start our work early.

The look of the two books is different, even though we have kept to the same overall structure. They envisage readers who are sitting in front of a computer and running version 8 of Stata at the same time. So they rarely provide output, because that would duplicate what is on the screen. We have tried to make this book usable even for those who do not yet have Stata, and have therefore included more screen shots of the dialogues and the output. Initial drafts of this book were based on Stata version 8. It is now updated to version 9.1.

We have used five datasets to illustrate the analyses, and these are all included on the CD, together with supporting information. The main four are from a survey of children born into poverty in Ethiopia, a livestock survey in Swaziland, a population study in Malawi and a socio-economic survey in Kenya. The fifth is a survey "game", based on a crop-cutting survey in Sri Lanka. We are very grateful to those who have encouraged us to provide this information, and we hope that readers will find that the datasets are of interest in their own right. They are described in Chapter 0 of Part 1 of the guide.

Chapter 11 Multiple responses

Multiple responses are a common feature of survey data when, to answer a single question, the respondent is allowed to "tick all boxes that apply" from a predetermined set of answers. We use data in the file **S_MultipleResponses.dta** described in Chapter 0.

11.1 Description of multiple responses questions

In the Swaziland livestock household questionnaire, question 9 asked if the household kept any livestock of 6 main species: cattle, sheep, goats, chickens, pigs and donkeys. Thus an individual could have kept up to six species of livestock. The interviewer had to fill in as many boxes as applied to the household, putting zero for the species not kept and the number of animals for those species that were kept. For example, the entry for household number 2 is:

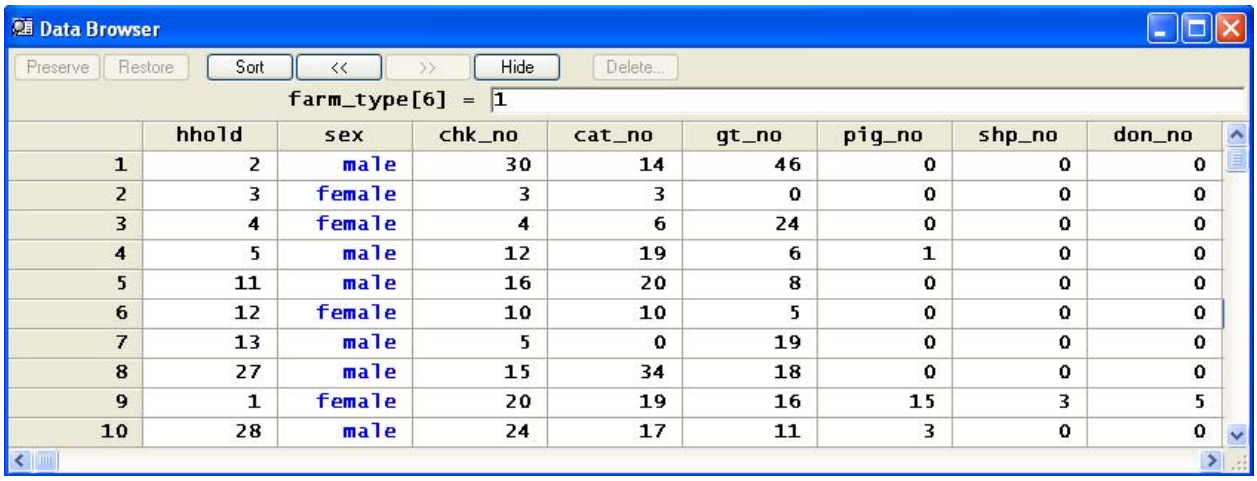
Q9. Livestock kept	(enter numbers in box)
Cattle	14
Sheep	0
Goats	46
Chickens	30
Pigs	0
Donkeys	0

If a household kept none of the 6 species mentioned, all the value recorded would be zero: such households are omitted from the dataset.

As shown in ***Fig. 11.1***, the **S_MultipleResponses** dataset has the following 8 variables: **hhold** [household unique identifier], **sex** [sex of the household head], **chk_no** [number of chickens], **cat_no** [number of cattle], **gt_no** [number of goats], **pig_no** [number of pigs], **shp_no** [number of sheep] and **don_no** [number of donkeys]. Open the **Stata** dataset with:

. use S_MultipleResponses, clear

Fig. 11.1 Browsing the data

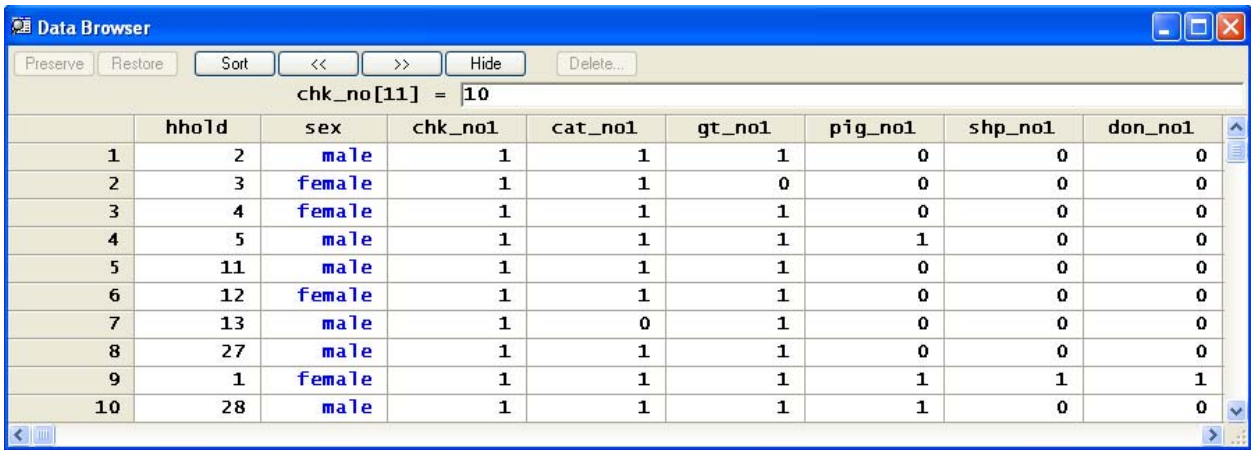


	hhold	sex	chk_no	cat_no	gt_no	pig_no	shp_no	don_no
1	2	male	30	14	46	0	0	0
2	3	female	3	3	0	0	0	0
3	4	female	4	6	24	0	0	0
4	5	male	12	19	6	1	0	0
5	11	male	16	20	8	0	0	0
6	12	female	10	10	5	0	0	0
7	13	male	5	0	19	0	0	0
8	27	male	15	34	18	0	0	0
9	1	female	20	19	16	15	3	5
10	28	male	24	17	11	3	0	0

This dataset in unusual because each variable is storing two pieces on information: if the livestock in question is kept and how many animals there are. For a typical multiple-response analysis, this type of storage would require recoding to multiple dichotomous variables in most packages, like SPSS, but it is not an issue in Stata.

Setting variables as multiple dichotomies is a much more common way of storing answers from multiple responses questions. It requires storing information as a set of 6 indicator variables, one for each major livestock species, with 0 if the species in question was not kept, or 1 if it was, see ***Fig. 11.2***.

Fig. 11.2 Using indicator variables to show presence of livestock on farm



	hhold	sex	chk_no1	cat_no1	gt_no1	pig_no1	shp_no1	don_no1
1	2	male	1	1	1	0	0	0
2	3	female	1	1	0	0	0	0
3	4	female	1	1	1	0	0	0
4	5	male	1	1	1	1	0	0
5	11	male	1	1	1	0	0	0
6	12	female	1	1	1	0	0	0
7	13	male	1	0	1	0	0	0
8	27	male	1	1	1	0	0	0
9	1	female	1	1	1	1	1	1
10	28	male	1	1	1	1	0	0

The document on the Stata website under the Data management FAQ link: "how do I deal with multiple responses?" http://www.stata.com/support/faqs/data/multresp.html gives a more detailed discussion of this topic.

11.2 Using an ADO file

There is no menu in Stata to deal with multiple responses, but fortunately, a user contributed ADO file can be downloaded both from the CD provided. It can also be downloaded from http://econpapers.hhs.se/software/bocbocode/S437201.htm]. Download both the **`mrtab.ado`** and **`mrtab.hlp`** files and save them in the **ADO/updates/m** folder of Stata [wherever it is installed in your PC]. Then run the **ADO** file from within the **DO Editor** window to compile the **mrtab** command. Next time you reload Stata, the mrtable command will be already available. Note that the earliest working for this command in version is 8.2.

If your Stata installation is set up correctly to update from the web [see Section 19.3] you can simply type:

. ssc install mrtab

This downloads and installs both ADO and HELP files.

11.3 Special nature of multiple responses

Suppose we want to know which percentage of households kept which type of animal. Since 6 columns store information from a single question, they must be summarized together in the same table. So we need a table that tallies only values larger than 0, and for computing percentages there are two denominators: one is the total number of respondents [cases, in Stata language], which is the length (number of rows) of each single column, here 454, and the other is the total number of responses, which is the total number of non-zero values over the 6 columns [1411 here]. The latter corresponds to the total number of responses given by all respondents.

Thus the number of types of animal per household is 1411/454=3.11, i.e. a household keeps 3 species on average.

For quick tabulation of multiple response questions it is advantageous to attach a common prefix to all 6 variables so they can be referred to collectively by using a wildcard: here we use q9. We also spell out in full the animal names.

. rename chk_no q9_chickens

. rename cat_no q9_cattle

….. and so on.

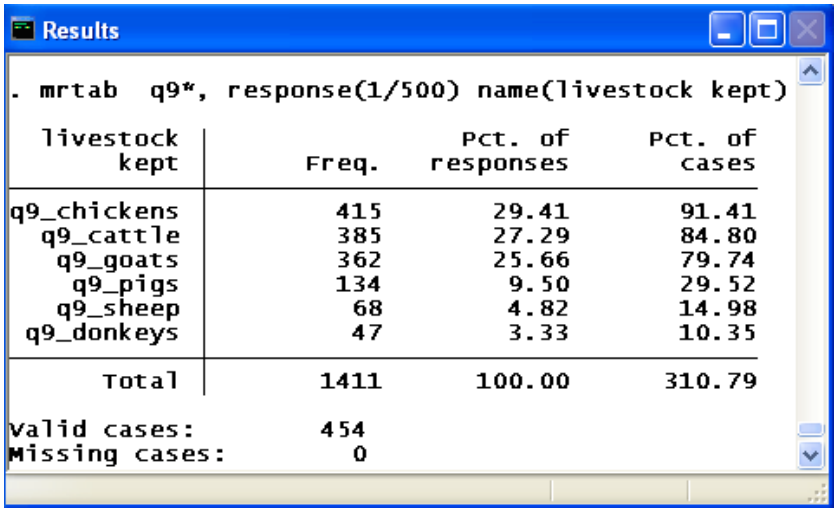
11.4 One-way tabulation

Assuming you have done this, you are ready for tabulating the 6 variables together using:

. mrtab q9*, response(1/500) name(livestock kept)

whose output is shown in ***Fig. 11.3***.

Fig. 11.3 Results from the mrtab command



livestock kept	Freq.	Pct. of responses	Pct. of cases
q9_chickens	415	29.41	91.41
q9_cattle	385	27.29	84.80
q9_goats	362	25.66	79.74
q9_pigs	134	9.50	29.52
q9_sheep	68	4.82	14.98
q9_donkeys	47	3.33	10.35
Total	1411	100.00	310.79

Valid cases: 454
Missing cases: 0

The response( ) option enables us to tally values larger than zero in a single group: the upper limits of the range should be set to the largest number across the 6 columns, found with:

. summarize q9*

The table in ***Fig. 11.3*** already has percentages for both denominators of responses and cases.

So, those households that keep livestock have 3 species on average, mainly chickens, cattle and goats, which are kept by 90%, 85% and 80% of the households respectively. Less than a third of households keep pigs and only 10% keep donkeys.

Note that the last column of percentages in ***Fig. 11.3*** adds to about 311%. This happens because a household can keep more than one type of livestock. The value 311% is not a useful summary for interpretation.

The results in ***Fig. 11.3*** happened to be in order of decreasing frequency. Has this been otherwise, the sort option of **mrtab** could have re-arranged the categories.

11.5 Two-way tabulation

Suppose it is of interest to investigate if the sex of the household head makes a difference to which species of livestock is kept.

This can be done with

. mrtab q9*, response(1/500) by(sex) name(livestock kept)

which tallies the counts separately for the two sexes, as shown in ***Fig. 11.4***.

Fig. 11.4 Summary of multiple responses separated by sex

Results

```
. mrtab  q9*, response(1/500) by(sex) name(livestock kept)

  livestock |  sex of household head
       kept |      male      female |     Total
------------+----------------------+----------
q9_chickens |       324          90 |       414
  q9_cattle |       307          77 |       384
   q9_goats |       296          65 |       361
    q9_pigs |       101          33 |       134
   q9_sheep |        53          15 |        68
 q9_donkeys |        37          10 |        47
------------+----------------------+----------
      Total |      1118         290 |      1408
      Cases |       357          96 |       453

Valid cases:         453
Missing cases:         0
```

There is one less valid case in ***Fig. 11.3*** than in the one-way table in ***Fig. 11.3***; this is because household number 30 had a missing value for sex. You can check this with:

. list if missing(sex)

Though the two totals at the bottom of the two-way table in ***Fig. 11.4*** are a useful reminder of the two denominators, the frequency counts in the body of the table are not helpful for comparing males and females. For a more informative tabulation omit the frequencies and give the column percentages with:

. mrtab q9*, response(1/500) by(sex) name(livestock kept) nofreq column

The resulting output is shown in ***Fig. 11.5***.

Fig. 11.5 Summary of multiple response percentages by sex

Results

```
. mrtab  q9*, response(1/500) by(sex) name(livestock kept) nofreq column

  livestock |  sex of household head
       kept |      male      female |     Total
------------+----------------------+----------
q9_chickens |     90.76       93.75 |     91.39
  q9_cattle |     85.99       80.21 |     84.77
   q9_goats |     82.91       67.71 |     79.69
    q9_pigs |     28.29       34.38 |     29.58
   q9_sheep |     14.85       15.63 |     15.01
 q9_donkeys |     10.36       10.42 |     10.38
------------+----------------------+----------
      Total |    313.17      302.08 |    310.82

Valid cases:         453
Missing cases:         0
```

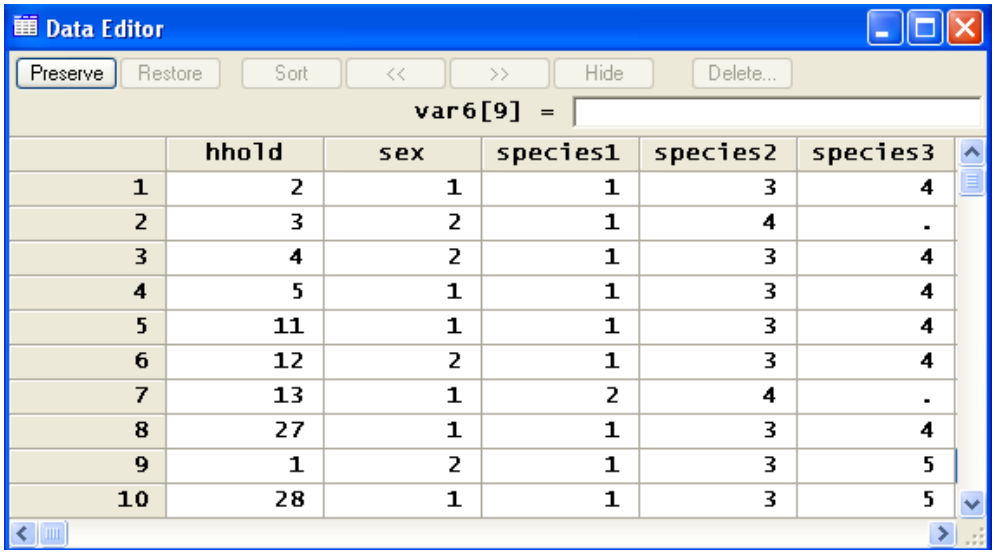
Fig. 11.5 shows that households whose head is female keep more chickens and pigs than male headed households. The opposite is true for cattle and goats. Hardly any difference is seen in the percentage of households keeping sheep and donkeys.

11.6 Polytomous variables

The **mrtab** command includes the option poly for dealing with another type of coding multiple responses, known as **`polytomous`** variables. This format is especially useful when the number of responses is limited to a subset of all possible answers.

Question 9 in the questionnaire asked for the rank of up to 3 most important species among the 6 mentioned. Each response is usually represented by a variable storing values from all available codes. For example, here there would be three variables ( ), each one with possible values 1 to 6, from cattle to donkeys, as shown in ***Fig. 11.6***.

Fig. 11.6 The three species regarded as most important (coded from 1 to 6)



	hhold	sex	species1	species2	species3
1	2	1	1	3	4
2	3	2	1	4	.
3	4	2	1	3	4
4	5	1	1	3	4
5	11	1	1	3	4
6	12	2	1	3	4
7	13	1	2	4	.
8	27	1	1	3	4
9	1	2	1	3	5
10	28	1	1	3	5

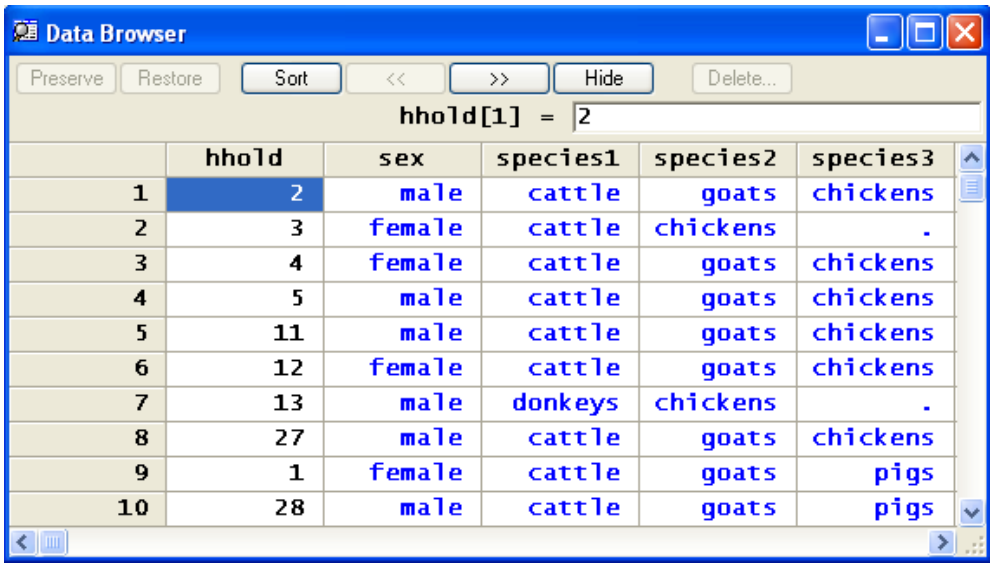
Notice that households 3 and 13 only kept 2 of the main species.

It is also more informative to attach value labels to all numeric codes as shown in ***Fig. 11.7***.

The commands are as follows:

```
. label define sexlab 1 male 2 female
. label values sex sexlab
. label define sp_label 1 cattle 2 donkeys 3 goats 4 chickens 5 pigs 6 sheep
. label values species1 sp_label
. label values species2 sp_label
. label values species3 sp_label
```

Fig. 11.7 The three most important species by name



	hhold	sex	species1	species2	species3
1	2	male	cattle	goats	chickens
2	3	female	cattle	chickens	.
3	4	female	cattle	goats	chickens
4	5	male	cattle	goats	chickens
5	11	male	cattle	goats	chickens
6	12	female	cattle	goats	chickens
7	13	male	donkeys	chickens	.
8	27	male	cattle	goats	chickens
9	1	female	cattle	goats	pigs
10	28	male	cattle	goats	pigs

The command

. mrtab species1 species2 species3, poly sort

Then gives equivalent results to those tabulated earlier in ***Fig. 11.3***.

Chapter 12 Regression and ANOVA

In this chapter, we show the use of Stata for fitting simple models, namely a simple linear regression model and a one-way analysis of variance (ANOVA) model. To illustrate, we use the rice survey example described in section 0.2.4 of this guide.

12.1 Fitting a simple regression model

We start by looking at a simple regression model. The aim of such a model is to investigate the relationship between two quantitative variables. Open the **`paddyrice.dta`** datafile and browse the data (see ***Fig. 12.1***). You will see that the rice yields are in a variable called **yield**, and the fertiliser amount used in the field that gave rise to this yield is in a variable called **fertiliser**. We will use Stata to explore how the amount of fertiliser affects the rice yields.

Fig. 12.1 Using the data browser for file paddyrice.dta

Data Browser

Preserve | Restore | Sort | << | >> | Hide | Delete...

yield[1] = 53.599998

	yield	village	field	size	fertiliser	variety
1	53.6	SABEY	3	2	2.5	OLD
2	44.6	SABEY	13	5	1.5	OLD
3	50.7	SABEY	4	5	3	OLD
4	33.6	SABEY	20	1.5	1	TRAD
5	62.1	SABEY	19	5	2.5	NEW
6	30.6	SABEY	10	4	1.5	TRAD
7	37.7	SABEY	8	4.5	2	OLD
8	24.3	SABEY	11	3.5	0	TRAD
9	56.8	SABEY	14	5	2	NEW
10	59.3	SABEY	12	4.5	2.5	OLD
11	40.4	KESEN	9	1.5	2	OLD
12	25.8	KESEN	8	7	0	TRAD
13	40.7	KESEN	1	2.5	1.5	OLD
14	27.6	KESEN	4	8	0	TRAD
15	48.7	KESEN	6	4.5	2.5	OLD
16	27	KESEN	3	2	.5	TRAD
17	19.1	KESEN	5	4	0	TRAD
18	26.3	NIKO	2	4.5	0	TRAD
19	24.7	NIKO	8	2.5	0	TRAD

First use **Graphics ⇒ Easy graphs ⇒ Scatter plot** to produce the graph in ***Fig. 12.2***. Then use **Statistics ⇒ Linear models and related ⇒ Linear regression** and complete the dialogue as shown in ***Fig. 12.3***. Pressing **OK** gives the output shown in ***Fig. 12.4***. Alternatively type the following commands:

```
. scatter yield fertiliser
. regression yield fertiliser
```

Fig. 12.2 Plot of yield versus fertiliser

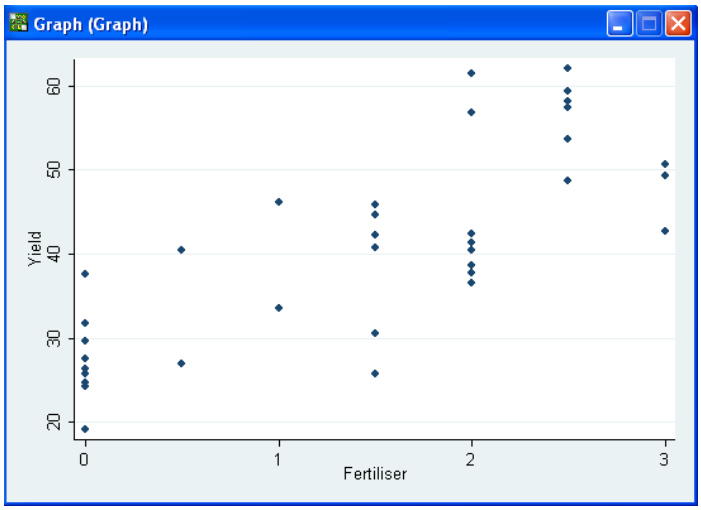


Fig. 12.3 Dialogue for regression

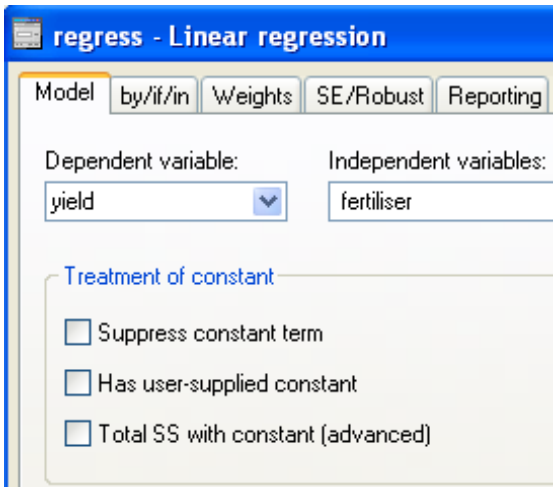


Fig. 12.4 Results of a regression of yield versus fertiliser

Results

```
. regress yield fertiliser

      Source |       SS       df       MS              Number of obs =      36
-------------+------------------------------           F(  1,    34) =   51.90
       Model |  2993.70059     1  2993.70059            Prob > F      =  0.0000
    Residual |  1961.03579    34  57.6775232            R-squared     =  0.6042
-------------+------------------------------           Adj R-squared =  0.5926
       Total |  4954.73638    35  141.563897            Root MSE      =  7.5946

------------------------------------------------------------------------------
       yield |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
  fertiliser |    8.948143   1.24203     7.20   0.000     6.424034    11.47225
       _cons |    27.65546   2.195619    12.60   0.000     23.19343     32.1175
------------------------------------------------------------------------------
```

From results of ***Fig. 12.4*** we see that the equation of the fitted regression line is:

yield = 27.7 + 8.9 * fertiliser

The fitted (predicted) yield values from this line can be saved in a variable called **fitted** using:

. predict fitted

The fitted line can then be displayed along with the raw data (see ***Fig. 12.5***) using:

. scatter yield fertiliser || line fitted fertiliser

An alternative is to use **Graphics ⇒ Easy graphs ⇒ Regression fit** and complete the dialogue as shown in ***Fig. 12.6***. Pressing **OK** gives the graph shown in ***Fig. 12.7***. The command generated by this menu sequence is:

. twoway (lfitci yield fertiliser) (scatter yield fertiliser)

The **lfitci** in the command above indicates that the fitted line would be shown along with the 95% confidence interval for the true value of the predicted mean yield.

Fig. 12.5 Plot of yield versus fertiliser with the fitted regression line

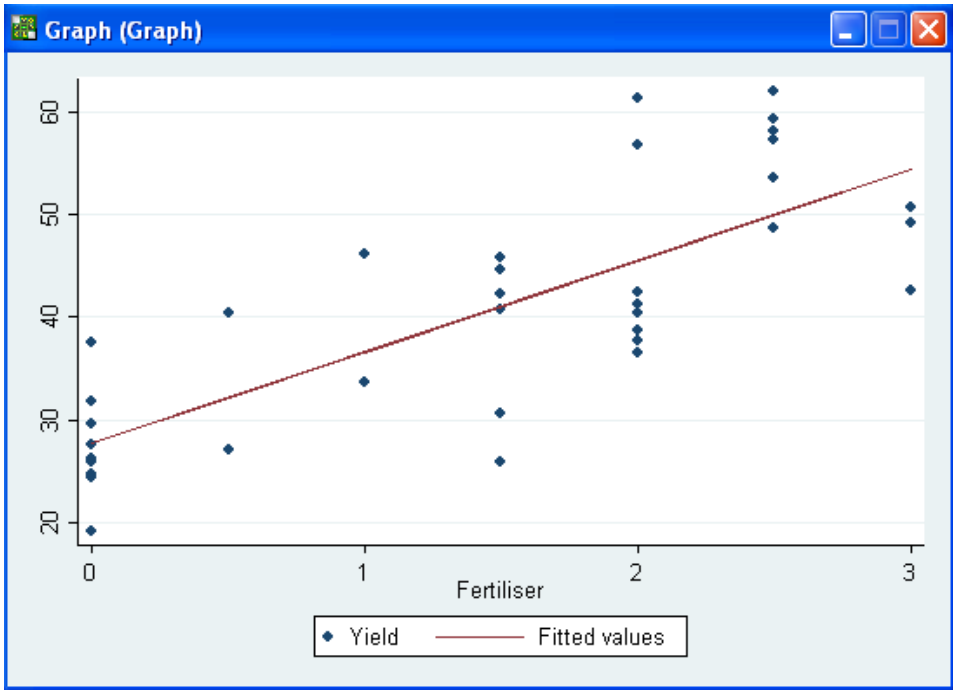


Fig. 12.6 Scatter, line and conf. limits

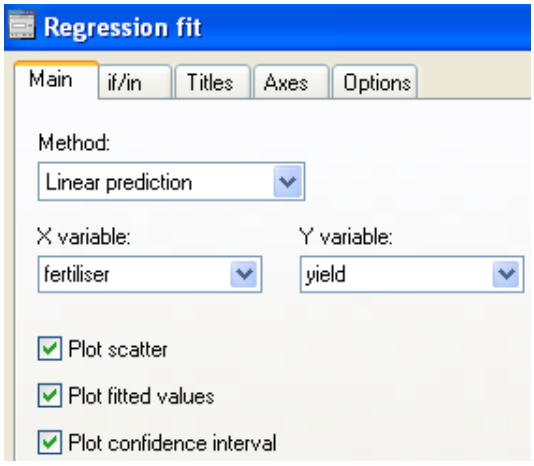
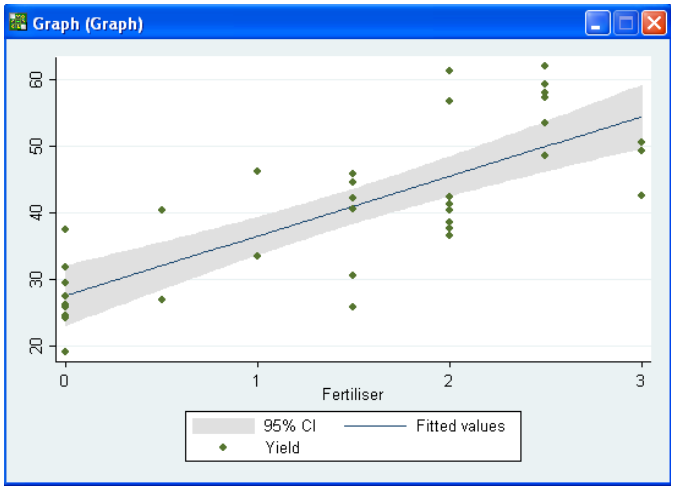


Fig. 12.7 Scatter plot with 95% confidence limits



12.2 Fitting a one-way analysis of variance (anova) model

In the paddy example above, it would also be of interest to investigate whether the mean yield of rice varies across the different varieties used. Try the following command to see how many varieties are grown by farmers visited during this survey.

. tab variety

In the output shown in the Results Window, “new” refers to a new improved variety, “old” refers to an old improved variety, while “trad” refers to the traditional variety used by farmers. The mean yields under each of these three varieties can be seen using the command:

. table variety, contents (mean yield sd yield freq)

The results are shown in ***Fig. 12.8***. Clearly the mean yield of the new variety is much higher than the mean yield of the other two varieties. But we would like to confirm that this is a real difference and not a chance result.

A statistical test, i.e. the one-way analysis of variance (anova) can be used for this purpose. Try

. oneway yield variety

The output from the above command is shown in ***Fig. 12.9***. The F-probability 0.0000 indicates clear evidence of a significant difference amongst the three variety means.

Fig. 12.8 Mean yields for each variety

Results

```
. table variety, contents(mean yield sd yield freq)
```

Variety	mean(yield)	sd(yield)	Freq.
NEW	59.6	2.554735	4
OLD	45.44118	7.125838	17
TRAD	30	6.518654	15

Fig. 12.9 One-way anova for comparing yields across varieties

Results

```
.
. oneway yield variety

                        Analysis of Variance
    Source              SS         df      MS            F     Prob > F
------------------------------------------------------------------------
Between groups      3527.81524      2   1763.90762     40.79     0.0000
 Within groups      1426.92113     33   43.2400344
------------------------------------------------------------------------
    Total           4954.73638     35   141.563897

Bartlett's test for equal variances:  chi2(2) =   3.1185  Prob>chi2 = 0.210
```

12.3 Using the *anova* command

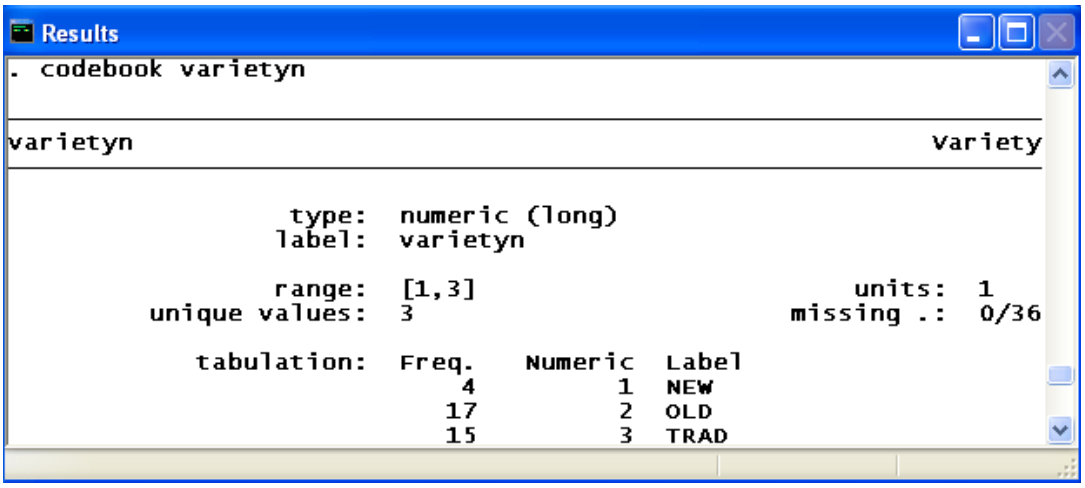
Another way to get the same results as from the ***oneway*** command above, is to use the ***anova*** command. However, this requires **variety** to be a numeric variable since in the data file, variety currently exists as a text variable. We can make variety into a new numerical variable using:

. encode variety, generate(varietyn)

. codebook varietyn

See ***Fig. 12.10*** to see the result.

Fig. 12.10 Codebook for numerical variable(codes) describing variety

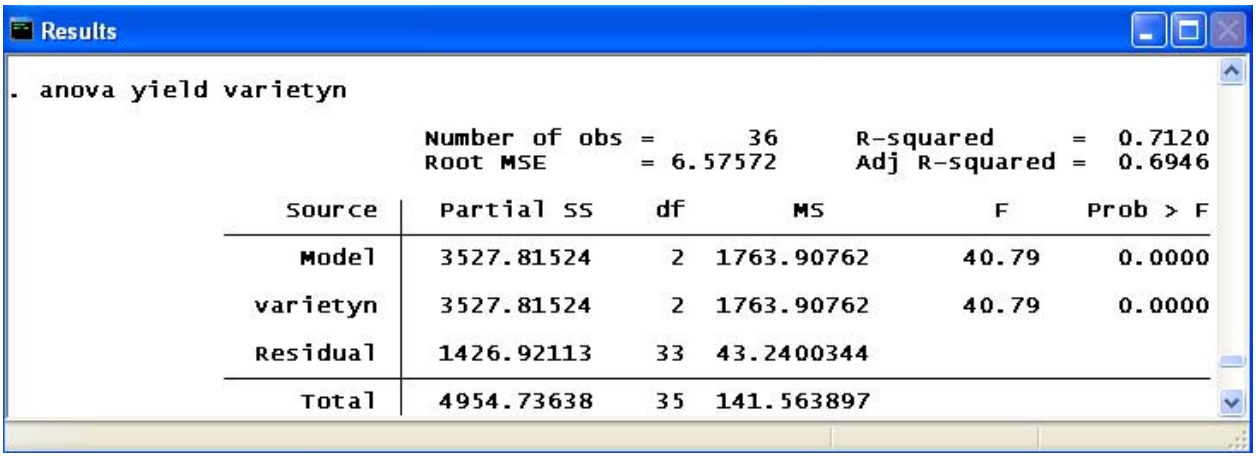


Now the anova command can be used as follows:

. anova yield varietyn

The output is in ***Fig. 12.11***. In this output, the "Model" line will contain all terms included in the ***anova*** command as potential explanatory factors that contribute to variability in yields. Here only one factor, namely **variety**, has been included. Hence the "Model" line coincides with results in the **varietyn** line.

Fig. 12.11 Results from use of anova command



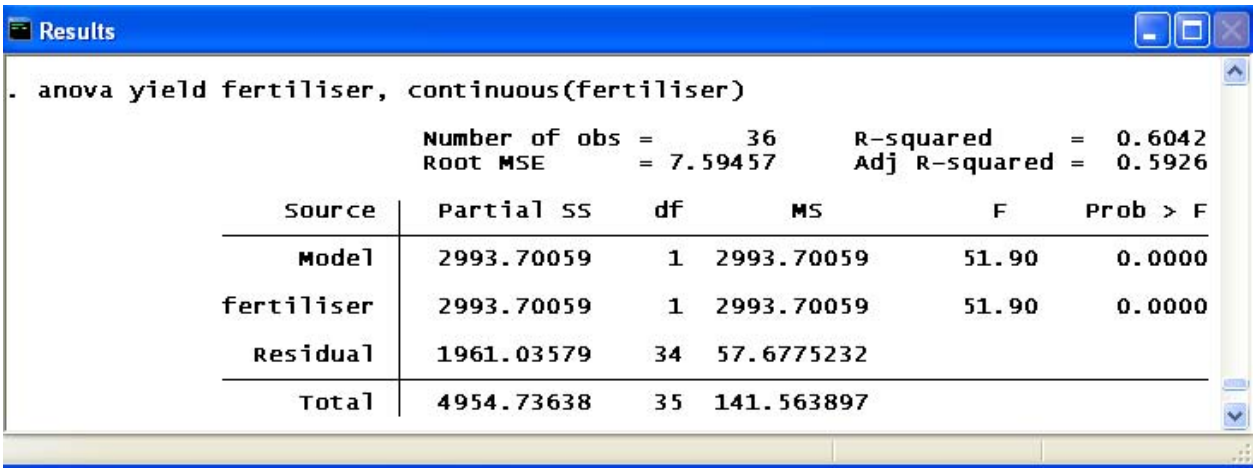
Note that the ***anova*** command can also be used to fit the simple linear regression model considered in section 12.1. However, the anova command expects all the explanatory variables to be categorical variables, and therefore if a quantitative variable such as **fertiliser** is used (to produce a simple linear regression model), then an option to the anova command must be used to indicate that fertiliser is a quantitative variable. So to produce the regression results shown in ***Fig. 12.4***, we must use the ***anova*** command as shown below.

. anova yield fertiliser, continuous(fertiliser)

The results are shown in ***Fig. 12.12***. The results coincide with those shown in ***Fig. 12.4***. The exact output in ***Fig. 12.4*** can also be produced using:

. anova yield fertiliser, continuous(fertiliser) regress

Fig. 12.12 Reproducing regression results using anova command



It is also possible to use the greater power of the anova command to investigate how well the simple linear regression model relating yield to fertiliser fits the data.

We saw in ***Fig. 12.5*** and ***Fig. 12.7*** that there were only 7 possible values for the amount of fertiliser applied, ranging from 0 to 3. This was because fertiliser had been measured in sacks to the nearest half-sack. The repeated observations at the same fertiliser level allow a check of the adequacy of the straight-line model, by seeing whether the departures from the line are more than random variation (pure residual). This 'pure' residual is the variability between the yields at exactly the same fertiliser level.

To do this, we first copy the fertiliser column into a new variable because we want to use the same numbers as both a variate and a categorical column. One way is to use

. generate fert = fertiliser

Then use **Statistics ⇒ Linear models and related ⇒ ANOVA ⇒ Analysis of variance and covariance**. Complete the resulting dialogue box as shown in ***Fig. 12.13***. Notice that we have opted for sequential sums of squares. Alternatively, type the command:

. anova yield fertiliser fert, continuous(fertiliser) sequential

The results are in ***Fig. 12.14***. There, the lack of significance of the extra `fert` term, with 5 degrees of freedom, implies insufficient evidence that we need more than a straight-line model.

Fig. 12.13 Dialogue for checking adequacy of regression model

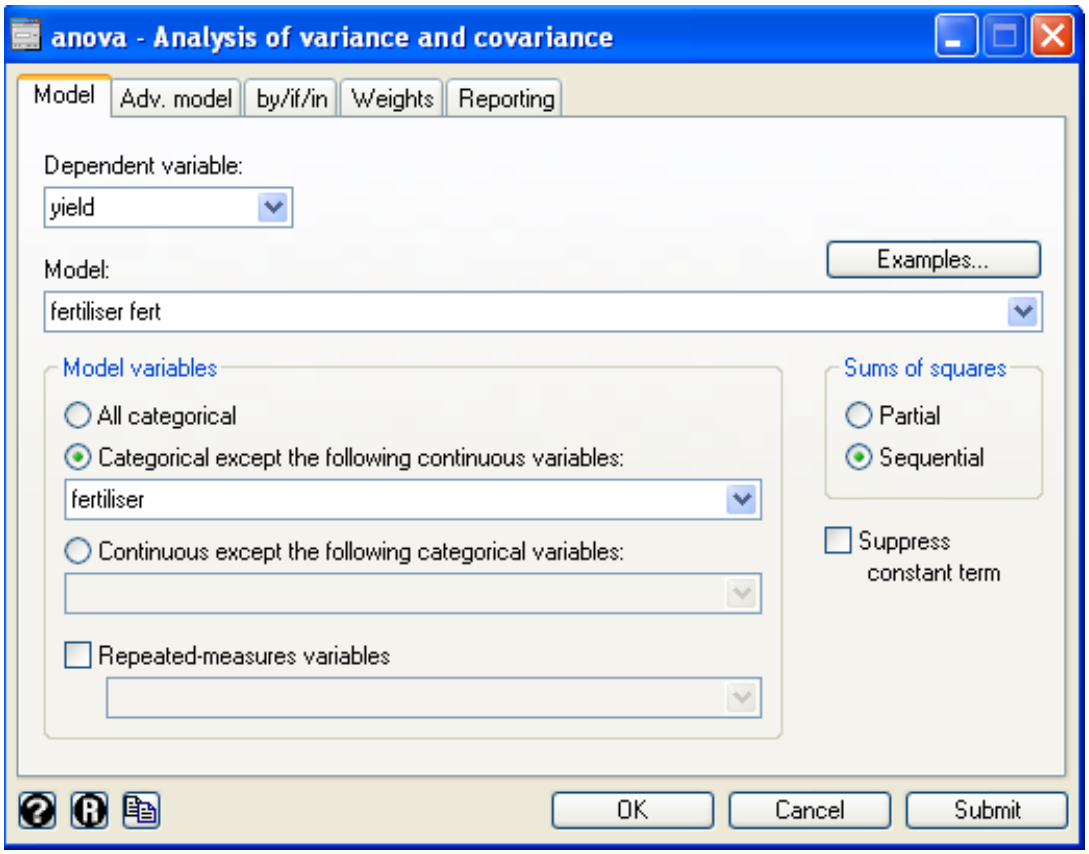
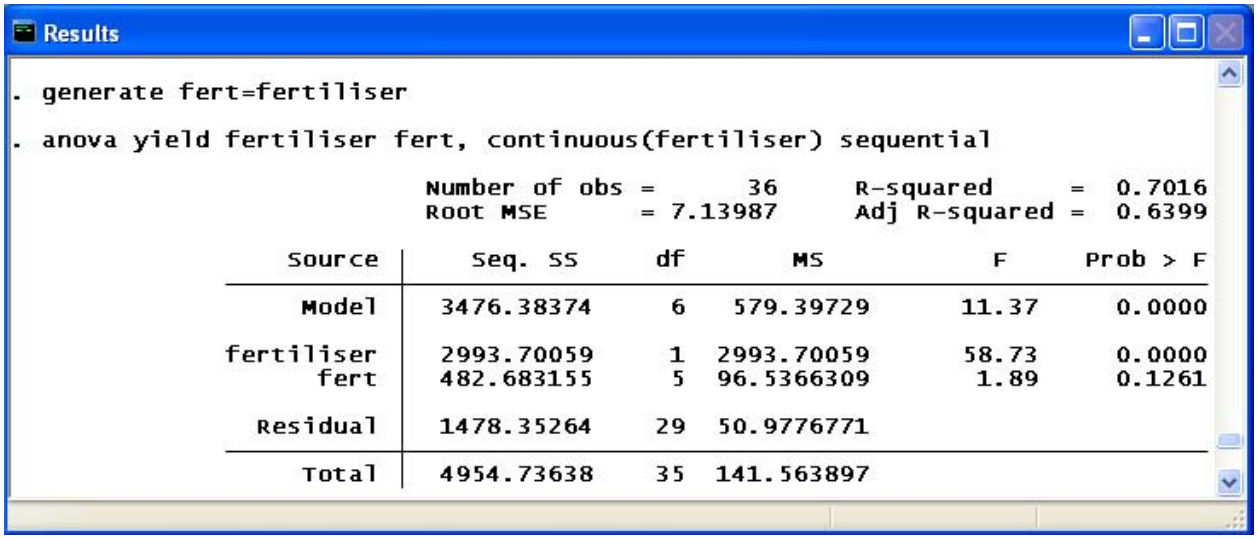


Fig. 12.14 Results for checking adequacy of regression model



In Chapter 16 we will further see the power of the ***anova*** command in fitting models including both continuous variables and categorical variables.

Chapter 13 Frequency and analytical weights

A key feature of Stata is the facility for using weights. One instance where weighting is needed for an analysis is when the data have already been summarised. In this chapter we illustrate the use of frequency weights for a regression analysis. In the next chapter we discuss the use of sampling weights.

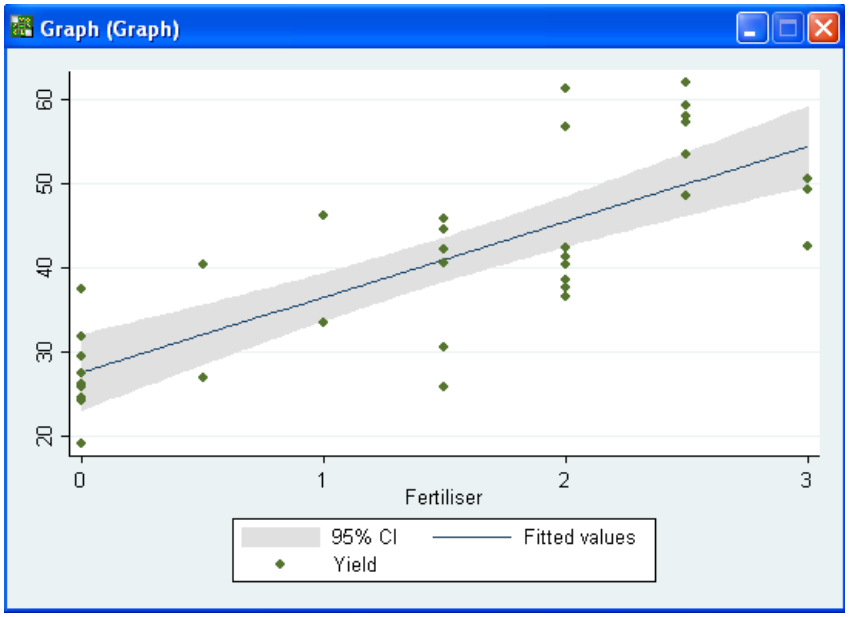
We again return to a simple linear regression model here, but it is primarily the data manipulation and general facilities in Stata for dealing with frequency weights that will be emphasised.

13.1 An example using a regression model

Begin by opening the **`paddyrice.dta file again`**, and as in Chapter 12, consider a simple regression model relating the rice yields to fertiliser inputs. Typing the following command will produce the output shown in ***Fig. 13.1***

. twoway (lfitci yield fertiliser) (scatter yield fertiliser)

Fig. 13.1 Scatter plot of yield by fertiliser with 95% confidence limits



The equation of the fitted line is obtained using:

. regress yield fertiliser

The results window (seen in ***Fig. 13.2***) gives the fitted line as

yield = 27.7 + 8.9 * fertiliser

13.2 Working with summarised data

Sometimes we may not have access to the individual data, and just have the means. We illustrate by generating the mean yields at each fertiliser level. Use **Data ⇒ Create or change variables ⇒ Other variable transformation commands ⇒ Make dataset of means, medians, etc**. Complete as shown in ***Fig. 13.3***. Also use the **Options** tab and specify that the data are to be collapsed over each level of fertiliser. This generates the command:

. collapse (mean) mnyield=yield (count) freq = yield, by(fertiliser)

Fig 13.2 Results from the regression of yield on fertiliser

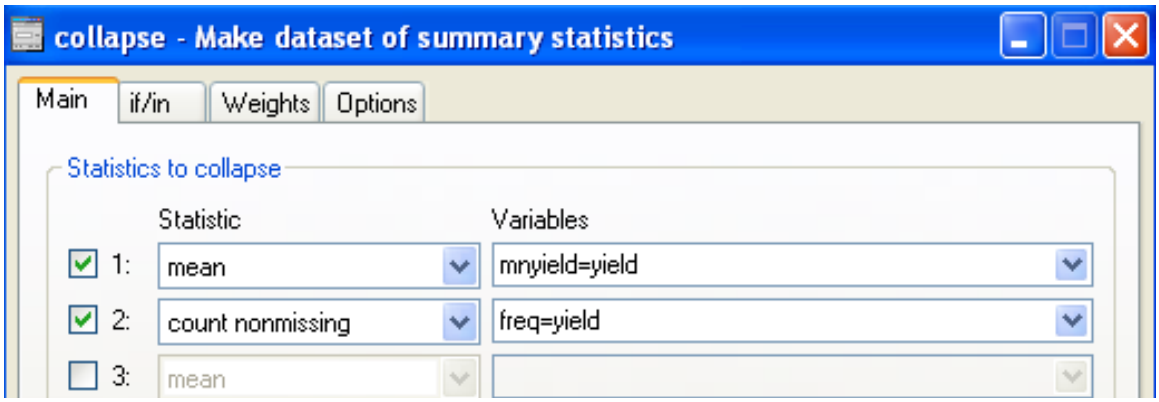
Results

```
. regress yield fertiliser
```

Source	SS	df	MS		
Model	2993.70059	1	2993.70059	Number of obs	= 36
Residual	1961.03579	34	57.6775232	F(1, 34)	= 51.90
				Prob > F	= 0.0000
				R-squared	= 0.6042
				Adj R-squared	= 0.5926
Total	4954.73638	35	141.563897	Root MSE	= 7.5946

yield	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
fertiliser	8.948143	1.24203	7.20	0.000	6.424034	11.47225
_cons	27.65546	2.195619	12.60	0.000	23.19343	32.1175

Fig 13.3 Summarizing results for each fertiliser level



The result is to clear the dataset with the raw data and replace it by one containing the means. If you use browse you see the new data are as shown in ***Fig. 13.4.***

Fig. 13.4 Browsing summarised data

Data Browser

Preserve | Restore | Sort | << | >> | Hide | De

freq[7] = 3

	fertiliser	mnyield	freq
1	0	27.42222	9
2	.5	33.7	2
3	1	39.9	2
4	1.5	38.28333	6
5	2	44.4125	8
6	2.5	56.53333	6
7	3	47.56667	3

Suppose you were not supplied with the raw data, but were given these summary values. Could you still estimate the effect of the fertiliser as above? We use the same route to examine the similarities and differences.

Again type the commands:

. twoway (lfitci yield fertiliser) (scatter mnyield fertiliser)

. regress mnyield fertiliser

Do you get the same line and the same confidence bounds as before?

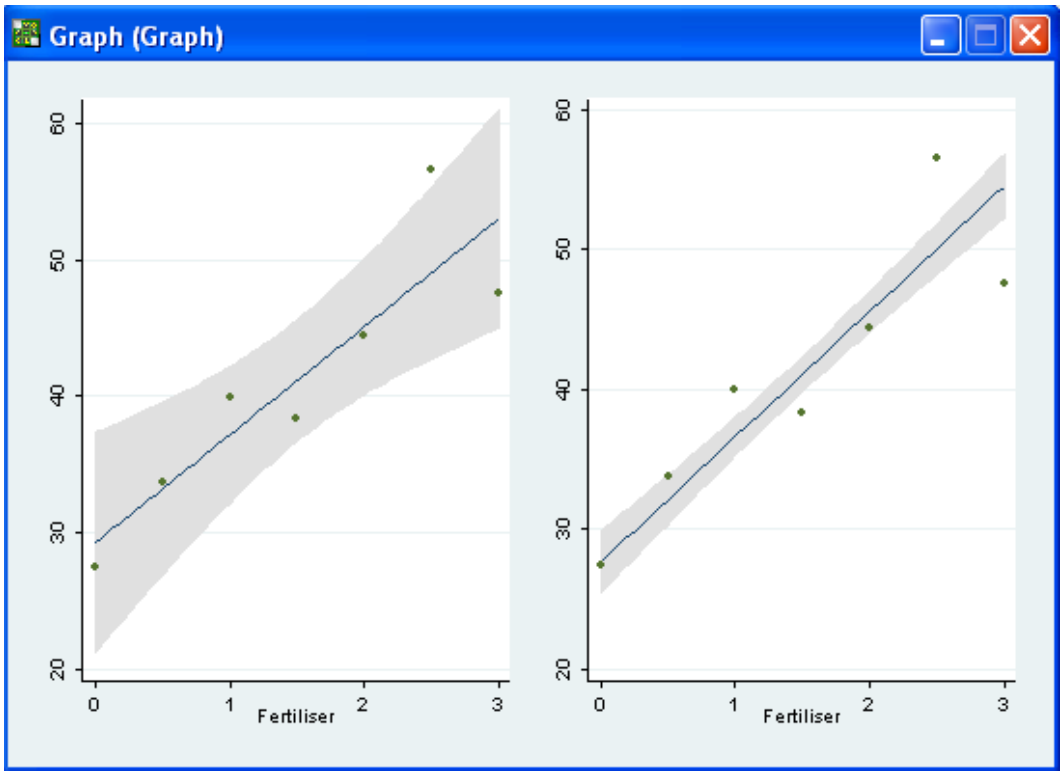
The answer is no, in both cases. The line (see the first pane of ***Fig. 13.5***) is not the same, because the analysis using the means has not taken any account of the different numbers of observations at the different fertiliser levels. The line would be the same if the replication had been equal at each fertiliser level.

We can rectify this aspect, though not from the menu. Recall the last `twoway` command and edit it to:

. twoway (lfitci mnyield fertiliser [fweight = freq]) (scatter mnyield fertilizer)

where the output is shown in the second pane of ***Fig. 13.5***.

Fig. 13.5 Fitted regression line without (left) and with (right) frequency weights



The change has been to do a weighted analysis, with the frequencies making up the weights. The equation of the fitted line is now the same as from the original data. We can check this by using the regression dialogue, i.e. using **Statistics ⇒ Linear models and related ⇒ Linear regression**, and filling the resulting dialogue box as shown in ***Fig. 13.6***.

Fig. 13.6 Main regression dialogue

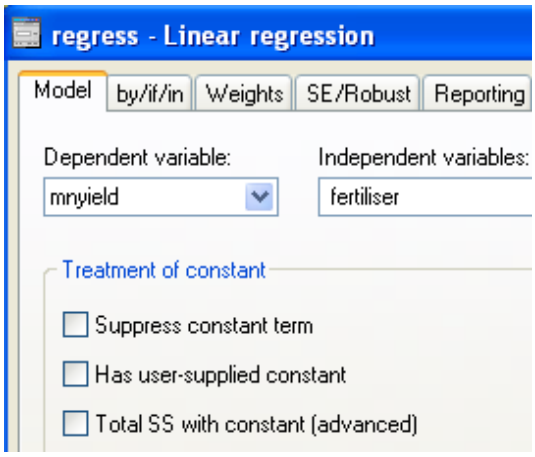
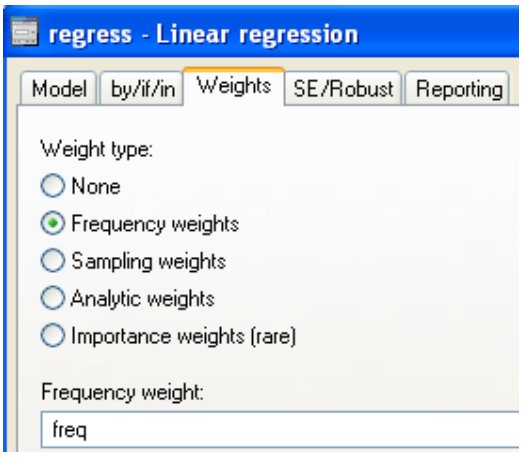
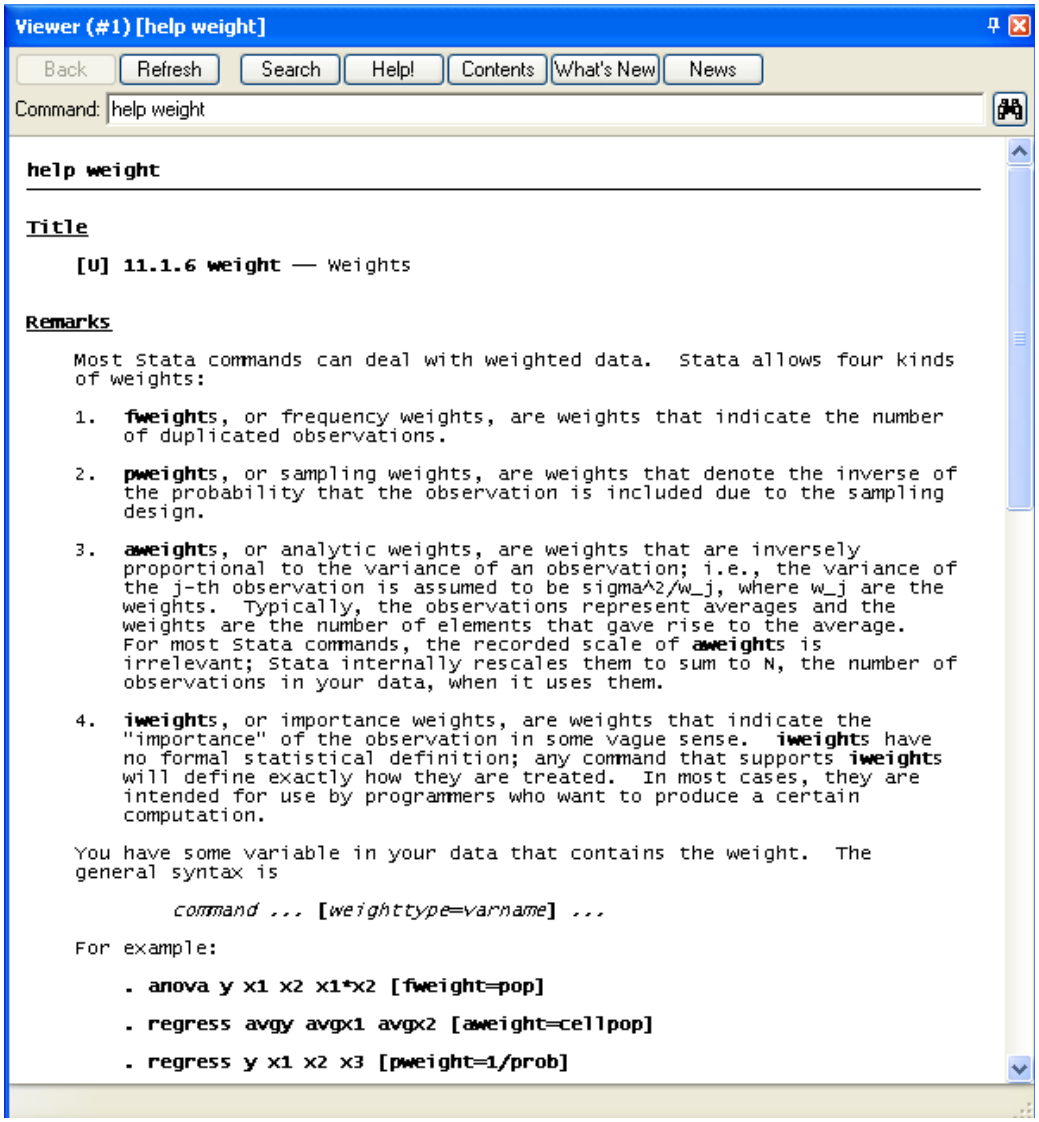


Fig. 13.7 Weights dialogue in regression



From the dialogue in ***Fig. 13.6*** we also use the tab called **weights**, which is on most of Stata's menus, and hence available with most commands. The resulting dialogue is shown in ***Fig. 13.7***, and we can use the **Help** button to learn more about the use of weights in Stata, see ***Fig. 13.8***.

Fig. 13.8 Description of weights in Stata



We see from ***Fig. 13.8*** that there are four types of weights we can use with Stata, and we will use two of these in this chapter. The first type is frequency weights, and they apply here. The second is analytic weights, and we will see that they are actually the most appropriate for the analyses in this chapter. We will consider sampling weights in Chapter 14.

Using the frequency weights generates the command:

. regress mnyield fertiliser [fweight=freq]

The results are in ***Fig. 13.9***, and can be compared with those from ***Fig. 13.2***.

Fig. 13.9 Regression results with summarized data using frequency weights

Results

```
. regress mnyield fertiliser [fweight = freq]

      Source |       SS       df       MS              Number of obs =      36
-------------+------------------------------           F(  1,    34) =  210.87
       Model |  2993.70025     1  2993.70025            Prob > F      =  0.0000
    Residual |  482.683277    34   14.196567           R-squared     =  0.8612
-------------+------------------------------           Adj R-squared =  0.8571
       Total |  3476.38353    35  99.3252436           Root MSE      =  3.7678

------------------------------------------------------------------------------
     mnyield |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
  fertiliser |    8.948142   .6161981    14.52   0.000     7.695877    10.20041
       _cons |    27.65546   1.089295    25.39   0.000     25.44175    29.86917
------------------------------------------------------------------------------
```

The equation is the same as we gave earlier using the full set of data, and that is a key result. So the graph on the right-hand side of ***Fig. 13.5*** gives the same equation, using the means, as we get from the original data. Comparing the ANOVA table in ***Fig. 13.9***, with the one given in ***Fig. 13.2***, we see that the model sum of squares is 2993.7 in both cases. So far, so good.

But the total sum of squares, of 3476, in ***Fig. 13.9***, with 35 degrees of freedom, is not the same as in ***Fig. 13.2***. It is lower. This is giving us a spurious impression of precision, which we can see visually, by comparing the width of the confidence band in the graph on the right of ***Fig. 13.5***, with the graph on the left.

If you replace the term **`fweight,`** by **`weight`**, in the command above, then Stata will use the type of weights that are usually most appropriate for a particular command. The results are in ***Fig. 13.10.***

Fig. 13.10 Regression results with summarized data using analytic weights

Results

```
. regress mnyield fertiliser [weight = freq]
(analytic weights assumed)
(sum of wgt is   3.6000e+01)

      Source |       SS       df       MS              Number of obs =       7
-------------+------------------------------           F(  1,     5) =   31.01
       Model |  582.108382     1  582.108382            Prob > F      =  0.0026
    Residual |  93.8550817     5  18.7710163           R-squared     =  0.8612
-------------+------------------------------           Adj R-squared =  0.8334
       Total |  675.963463     6  112.660577           Root MSE      =  4.3326

------------------------------------------------------------------------------
     mnyield |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
  fertiliser |    8.948142   1.606848     5.57   0.003     4.817607    13.07868
       _cons |    27.65546   2.840533     9.74   0.000     20.35364    34.95728
------------------------------------------------------------------------------
```

We see that Stata assumes analytic weights. The analysis shows that the equation of the line is the same as before, which is a relief. The degrees of freedom in the Analysis of Variance table, are now what we would expect. We have 7 data points and hence a total of 6 degrees of freedom. The regression line is estimating a single slope, and therefore has one degree of freedom. This leaves five degrees of freedom for the residual.

The sum of squares for the model in ***Fig. 13.10*** is 582.1. To see the correspondence with ***Fig. 13.2***, note that we have here 7 observations, and each one is a mean. Earlier there were 36 individual observations. Multiply 582.1 by 36/7 gives 2993.7, as before (see ***Fig.13.9*** and ***Fig. 13.2***.). The residual term 93.855, when multiplied by 36/7, gives 482.7. The same applies to the total.

So, with the analytic weights we get the right equation, and test the goodness of fit against the variability of the means about the line. This is the best we can do with the means, because we no longer have the raw data to provide the pure error term. Hence to complete the analysis, you may wish to redo the graph with the changed weights, i.e.

. twoway (lfitci mnyield fertiliser [aweight = freq]) (scatter mnyield fertiliser)

If you wish, you can replace **`aweight`** by just **`weight`**, in the command, because Stata will then assume analytic weights are needed. We will use **`aweight`** for the weighted analysis in the rest of this chapter.

13.3 Summaries at the village level

Sometimes the raw data are not provided for the analyses. The volume may be too great, or the individual records may not respect the confidentiality that was promised when the data were collected. Instead summaries are given at a higher level. We illustrate with the rice survey dataset again. We look first at the individual observations and then summarise to the village level.

Open the **`paddyrice.dta`** file, and summarise the yields and quantities of fertiliser applied. The results are in ***Fig. 13.11***.

Fig. 13.11 Summary statistics for yield and fertiliser

Results

```
. summarize yield fertiliser
```

Variable	Obs	Mean	Std. Dev.	Min	Max
yield	36	40.58056	11.89806	19.1	62.1
fertiliser	36	1.444444	1.033564	0	3

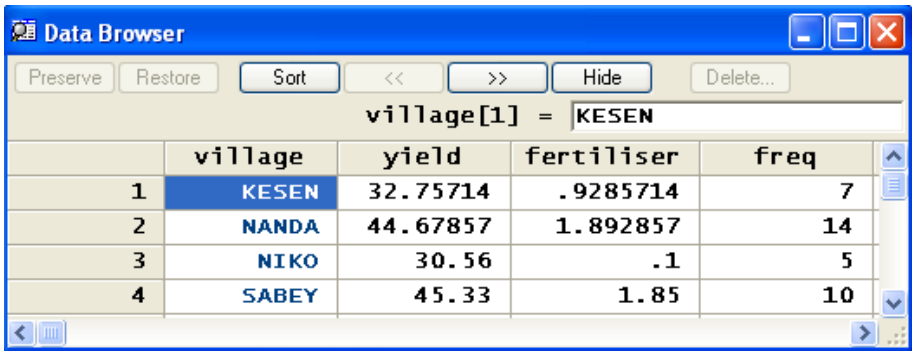
The results are simple to interpret. For example we see that the mean yield was 40.6, and the best farmer had a yield of 62.1. (The yields are in 1/10 of a ton.)

Now we summarise to the village level, prior to making the summary data available. We can use the menus as described before, or type:

. collapse (mean) yield fertiliser (count) freq=yield, by(village)

The resulting summaries are shown in ***Fig. 13.12***. They are the data we want to use for further analyses.

Fig. 13.12 Browsing data summarised by village



	village	yield	fertiliser	freq
1	KESEN	32.75714	.9285714	7
2	NANDA	44.67857	1.892857	14
3	NIKO	30.56	.1	5
4	SABEY	45.33	1.85	10

We start by summarising these data in the same way as the individual observations above, though including weights. The results are shown in ***Fig. 13.13.***

Fig. 13.13 Data summaries using analytic weights

Variable	Obs	Weight	Mean	Std. Dev.	Min	Max
yield	4	36	40.58055	7.17811	30.56	45.33
fertiliser	4	36	1.444444	.7542551	.1	1.892857

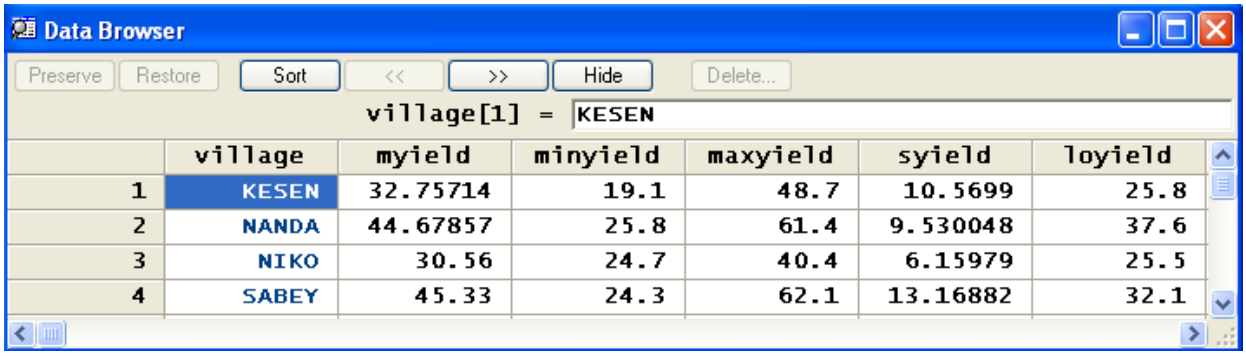
The means are as before, but how should we interpret the standard deviation, and the minimum and maximum. Here 30.6 is the minimum of the means and 45.3 is the maximum. So they represent the average yield in the villages with the lowest and highest averages. Similarly the standard deviation is an indication of the spread of averages over the different villages, and not the spread of individual observations.

The main advantage of the collapsing process is that it allows the resulting information to be combined with any further information existing at the village level.

If necessary it is also possible to collapse the data to the village-level, and still retain information about individual farmers, but we must request this information when we summarise the data.

To illustrate, open the original **`paddyrice.dta`** file again. The same collapse command or dialogue, used earlier, can be used to produce summaries other than the mean. For example in ***Fig. 13.14*** we show the village-level information that includes the mean yield again, but also the minimum value in each village (e.g. 19.1 for Kesen), the maximum, the standard deviation of the within-village yields, and also some percentiles. For example, in ***Fig. 13.14*** we have named the 20th percentile in each village as **`loyield`**. So, in village Kesen, 20% of the farmers had a yield of less than 25.8.

Fig. 13.14 Summaries by village for several summary statistics



	village	myield	minyield	maxyield	syield	loyield
1	KESEN	32.75714	19.1	48.7	10.5699	25.8
2	NANDA	44.67857	25.8	61.4	9.530048	37.6
3	NIKO	30.56	24.7	40.4	6.15979	25.5
4	SABEY	45.33	24.3	62.1	13.16882	32.1

Thus, when data are summarised from plot to village level, decisions have to be made regarding the summary measure to use for quantitative measurements like the yield. The appropriate summary measure to use depends on the objectives of the analysis.

13.4 Categorical data and indicator columns

There are some summaries that are not given directly with the collapse dialogue and command. For example suppose a low yield was defined as a yield of less than 30 units. We would like the count or perhaps the proportion of farmers in each village with less than this yield. This is the 'partner' to the percentiles that are given in ***Fig. 13.14***. In that case we fixed the specific percentile we needed (the 20th percentile) and found that this value was 25.8 in one of the villages. Now we wish to do the reverse, i.e. fix the yield quantity, and find the percentage of farmers getting yields lower than this quantity.

Re-open the `paddyrice.dta` file again. As usual, if what is required cannot be done in one step, then it usually requires an additional command. Type

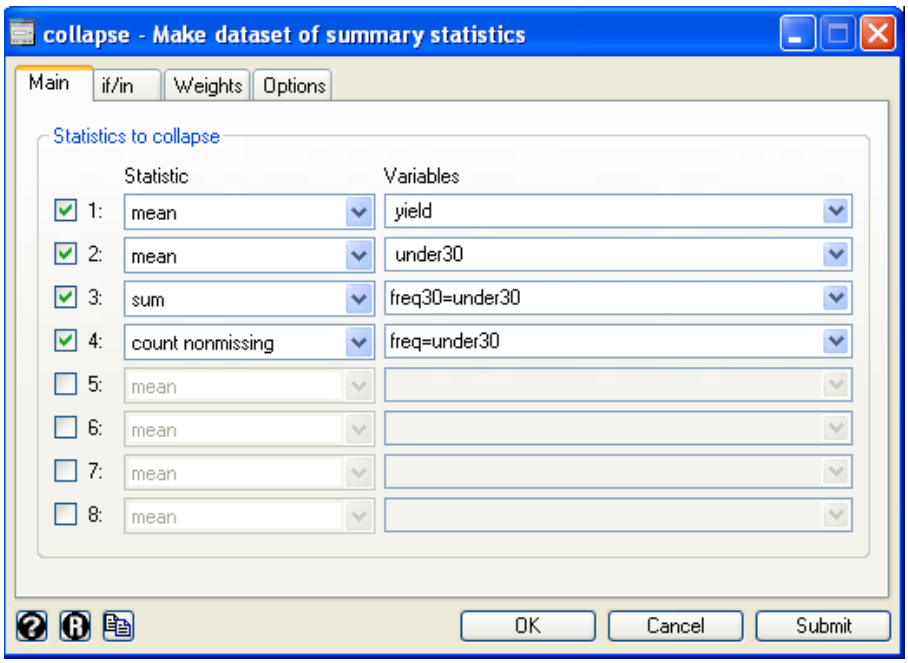
. gen under30=(yield<30)

Browse the data to see what the variable **under30** looks like. You will notice it is an "indicator" column, i.e. it has the value 1 when the corresponding yield is < 30, and zero otherwise.

Now use the dialogue as shown in ***Fig. 13.15***, or type the command directly as:

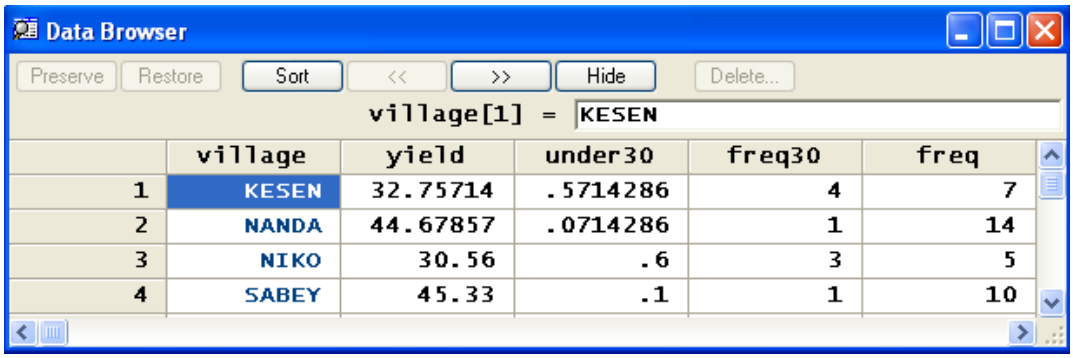
. collapse (mean) yield under30 (sum) freq30=under30 (count) freq=under30, by(village)

Fig.13.15 Summaries over farms with yields < 30



The results are shown in ***Fig. 13.16***. As can be seen from the third column, the mean of an indicator column gives the proportion of times the value is true, i.e. the yield is under 30. For illustration we have chosen to give both the count and the proportion. In practice we would usually just give the count, see the column `freq30` in ***Fig. 13.16***, because the proportion can then be calculated later. For example, in the first row of ***Fig. 13.16*** we see that 0.57 = 4 / 7.

Fig. 13.16 Browsing summaries for farms with yields < 30



	village	yield	under30	freq30	freq
1	KESEN	32.75714	.5714286	4	7
2	NANDA	44.67857	.0714286	1	14
3	NIKO	30.56	.6	3	5
4	SABEY	45.33	.1	1	10

13.5 Collapsing and use of anova

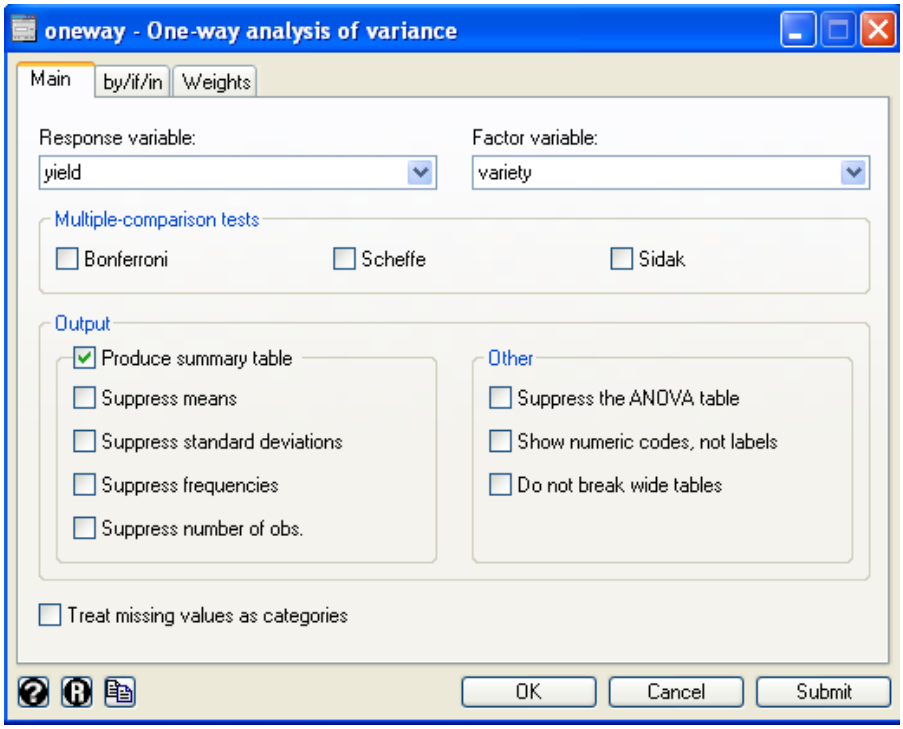
In Sections 13.4 and 13.2 we have concentrated largely on summarising the yields at the village level. But there is also other information. Open the **`paddyrice.dta`** file again, and this time look also at the information on the variety of rice used. This information may be of interest in its own right, or because we feel the yields might depend to some extent on the variety grown.

These two aspects may be linked, in that if there is no effect of variety on yields, then we do not wish to consider this aspect further. If there is an effect, then we would like to know the number, or proportion of farmers in each village that grow the improved varieties.

Use **Statistics ⇒ Linear models and related ⇒ ANOVA ⇒ One-way ANOVA** and complete the dialogue as shown in ***Fig. 13.17.*** Include ticking the option to produce a summary table. Alternatively type

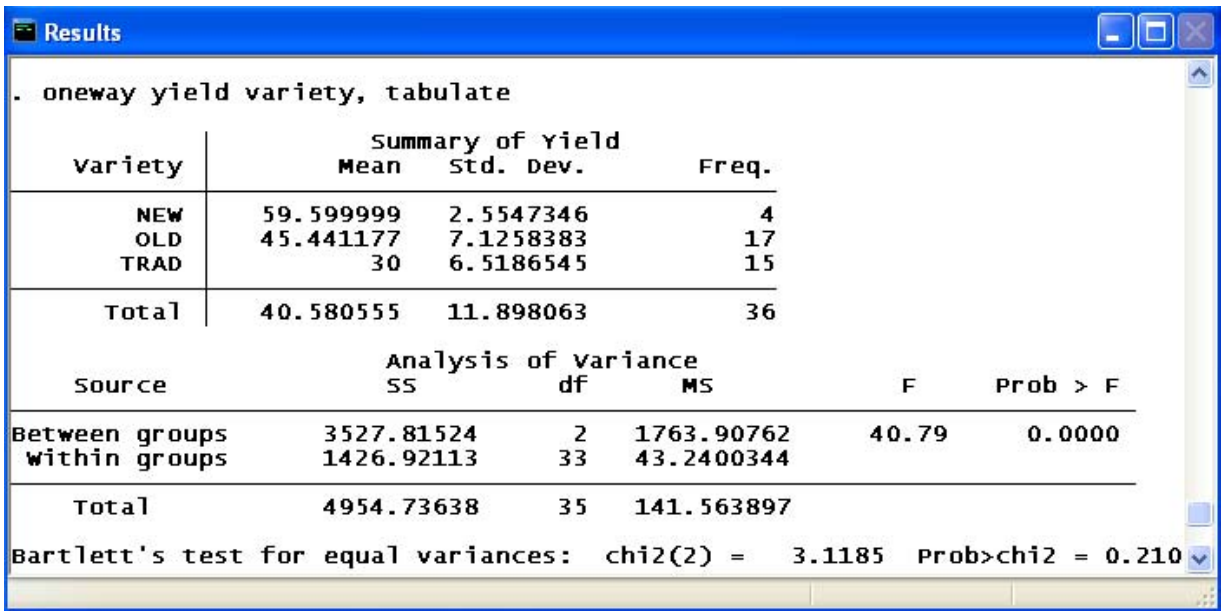
. oneway yield variety, tabulate

Fig. 13.17 ANOVA dialogue for comparing yields across varieties



The results are in ***Fig. 13.18*** and indicate a clear difference between the three varieties.

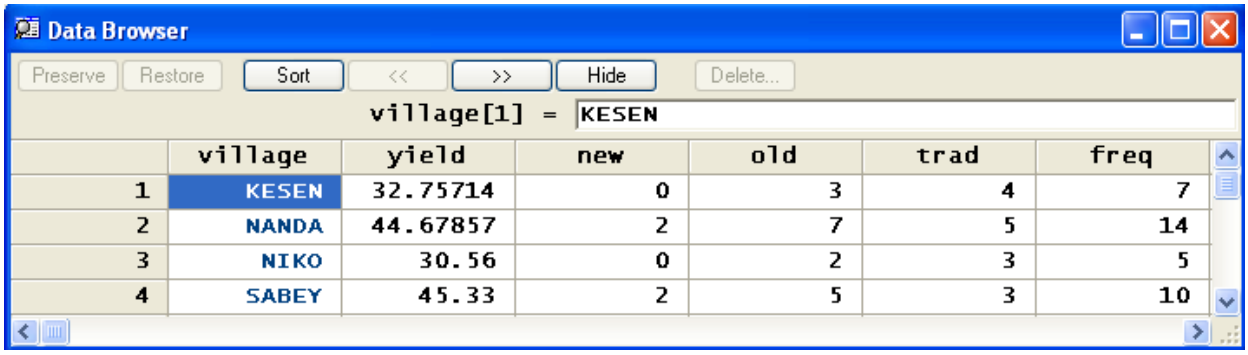
Fig. 13.18 Output from anova results



Suppose you now wish to summarise the data to the village level. We can just include a summary of the number of farmers in each village who grow each variety. For example

. gen trad=(variety==”TRAD”)
. gen old=(variety==”OLD”)
. gen new=(variety==”NEW”)
. collapse yield (sum) new old trad (count) freq=yield, by (village)

Fig. 13.19 Data summaries by village with numbers of farmers growing each variety



	village	yield	new	old	trad	freq
1	KESEN	32.75714	0	3	4	7
2	NANDA	44.67857	2	7	5	14
3	NIKO	30.56	0	2	3	5
4	SABEY	45.33	2	5	3	10

The resulting summary information allows some discussion still of the possible effect of variety. For example the two villages with higher mean yields are those where the new variety is used and where a smaller proportion of the farmers use the variety **TRAD**. But the clear message from ***Fig. 13.18*** is now very diluted.

An alternative is to keep the information separate for the different levels of the categorical column. Instead of the commands above, return to the main **`paddyrice.dta`** file, and try

. collapse yield (count) freq=yield, by (village variety)

The new feature is that we are collapsing by two category columns, namely both village and variety. As there are four villages and three varieties, you might expect there to be 12 rows of data. However, if you now use browse, you find there are only 10 rows. This is because two of the villages have no farmers who use the variety **`NEW`**.

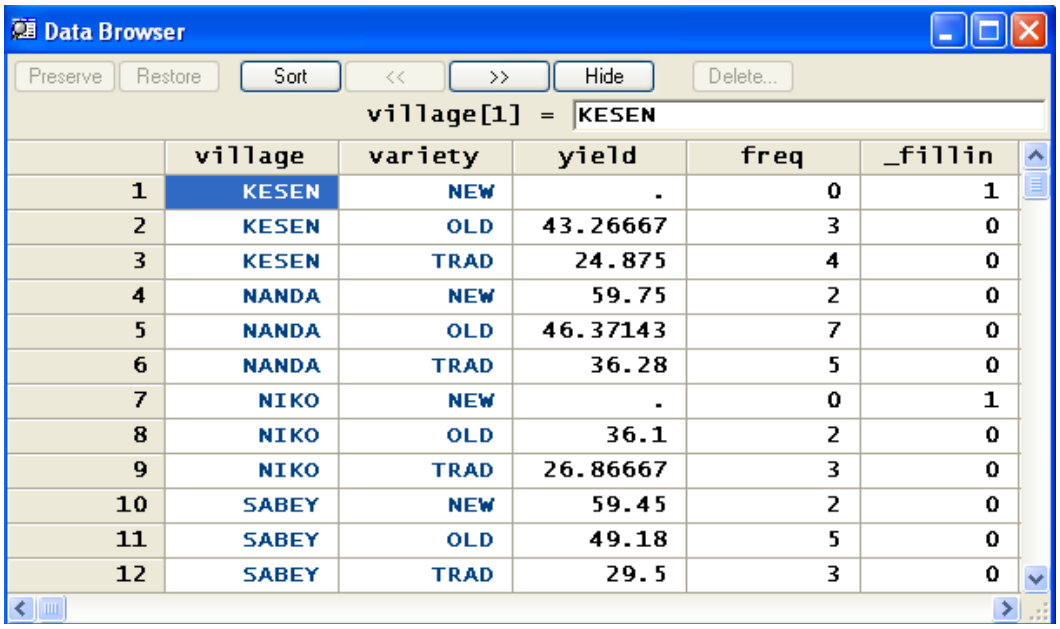
If you would like the 12 rows, then use the two commands below, or use the menu options, **Data ⇒ Create or change variables ⇒ Other variable transformation commands ⇒ Rectangularize dataset**, specifying variables **`village`** and **`variety`** in the resulting dialogue, followed by the second command below.

. fillin village variety

. mvencode freq if _fillin==1, mv(0)

The results are in ***Fig. 13.20***.

Fig. 13.20 Browsing data by village and variety



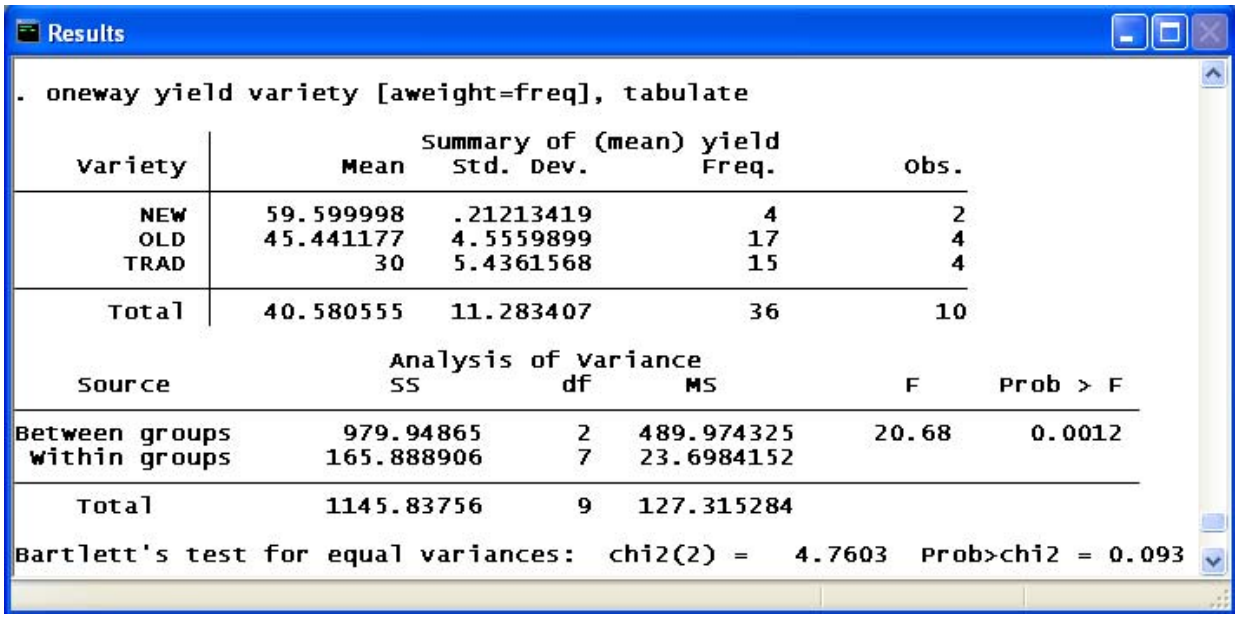
	village	variety	yield	freq	_fillin
1	KESEN	NEW	.	0	1
2	KESEN	OLD	43.26667	3	0
3	KESEN	TRAD	24.875	4	0
4	NANDA	NEW	59.75	2	0
5	NANDA	OLD	46.37143	7	0
6	NANDA	TRAD	36.28	5	0
7	NIKO	NEW	.	0	1
8	NIKO	OLD	36.1	2	0
9	NIKO	TRAD	26.86667	3	0
10	SABEY	NEW	59.45	2	0
11	SABEY	OLD	49.18	5	0
12	SABEY	TRAD	29.5	3	0

To show that this has still kept some of the information on the effect of the different varieties, we repeat the oneway analysis of variance on the summary data, using the frequencies as the weights, i.e.

. oneway yield variety [aweight=freq], tabulate

The output is in ***Fig. 13.21***. We see the means are as before, see ***Fig. 13.18***. The terms in the analysis of variance table are interpreted in exactly the same way as for the regression, described in Section 12.1. For example if we take the sum of squares for the groups, of 979.9, in ***Fig. 13.21*** and calculate 979.9*36/10, we get 3528, i.e. the “Between groups” SS shown in ***Fig. 13.18***.

Fig. 13.21 Results of ANOVA with analytic weights



13.6 In conclusion

In this chapter we have seen that it is easy to move data up a level from the plot to the village level. This is a common requirement in processing survey data and applies over many levels in real surveys. For example a national survey may include information at region, district, village and household level.

Whether summaries are effective depends on the objectives. Often we will find that objectives related to estimating totals or counts can safely be summarised, while those related to examining relationships need to be considered more carefully.

For example, with the survey considered in this chapter, suppose we also have information on the support to farmers by extension staff, and this is supplied at a village level, then it would be useful to summarise some of the individual data to the same village level in order to assess the impact of the support from extension staff.

Of course four villages is too few, but questions about how much difference an extension worker has made would naturally be assessed at the village level. Unravelling the effect of these differences from the farmers' point of view, for example in variety and fertiliser use, would still be done at the individual level. Thus, when looking at relationships, we will often find that our problem needs to be tackled at multiple levels, depending on the question.

Moving up from the individual to the village level has implied that subsequent analyses may have to be weighted. We have seen that Stata handles weighted analyses with ease. This is one of the strengths of the software. In the next chapter we will look at the facilities in Stata for handling sampling (probability) weights.

We have also looked at two simple models to start our understanding of how the rice yields relate to the inputs. In Sections 12.1, 13.1 and 13.2 we examined the relationship between yields and fertiliser, and in Sections 12.2 and 13.5 we looked at the relationship between yields and variety. Both aspects seem important. This is only the start of the modelling, because the two aspects may not be independent. For example the farmers who use the **NEW** variety all apply fertiliser, so we have to unravel the way both aspects, and possibly other variables interact. This is considered further in Chapter 16.

Chapter 14 Computing sampling weights

The purpose of this chapter is to show how sampling weights can be used to take account of the sampling structure when estimating population characteristics in household and other surveys. Data from the Malawi Ground Truth Investigation Study (GTIS) is used for this purpose. One of the objectives of this study was to estimate the size of the rural population of Malawi. The background to this study is as follows.

The census in 1998 estimated the rural population of Malawi as 1.95 million households and 8.5 million people. An update of this estimate was needed because registration of rural households for receiving a "started-pack" of seed and fertiliser (SPLU) in 1999 gave an unrealistic estimate of 2.89 million households, and hence about 12.6 million people. The GTIS survey aimed to provide an independent estimate of the size of Malawi's rural population.

14.1 The GTIS sampling scheme and the data

The GTIS covered all 8 Agricultural Development Divisions (ADDs) of Malawi. A minimum of 3 Extension Planning Areas (EPAs) were visited in each ADD (with one or two more EPAs added to the larger ADDs), giving a total of 30 EPAs. Two villages were selected in each EPA, resulting in a total of 60 villages. The selection of EPAs within ADD and of villages within EPA were done at random. This was thus a two-stage stratified sampling scheme, with ADD as strata, EPA as primary sampling units and villages as secondary sampling units.

Data concerning the number of households enumerated by GTIS, and additional information about the ADDs, EPAs and villages, are found in the file **`M_village.dta`**. Part of the data can be seen in ***Fig. 14.1***.

Fig. 14.1 Browsing the data from the GTIS survey

Data Browser

Preserve | Restore | Sort | << | >> | Hide | Delete...

ADD[1] = 1

	ADD	ADD_EPA	ADD_vill	ADD_hh	EPA	EPA_visit
1	Blantyre	27	3119	758757	Masambanjati	4
2	Blantyre	27	3119	758757	Masambanjati	4
3	Blantyre	27	3119	758757	Mulanje Boma	4
4	Blantyre	27	3119	758757	Mulanje Boma	4
5	Blantyre	27	3119	758757	Ntonda	4
6	Blantyre	27	3119	758757	Ntonda	4
7	Blantyre	27	3119	758757	Tamani	4
8	Blantyre	27	3119	758757	Tamani	4
9	Blantyre	27	3119	758757	Waruma	4
10	Blantyre	27	3119	758757	Waruma	4
11	Karonga	9	655	77218	Chitipa North	3
12	Karonga	9	655	77218	Chitipa North	3

A description of the variables in the dataset may be obtained with the following Stata commands:

```
. use M_village
. describe
```

These commands give the results shown in ***Fig. 14.2***. A list of the ADDs, the number of EPAs in each ADD, and the numbers visited, are shown in ***Fig. 14.3***, produced by using the command:

```
. table ADD if village==1, contents(mean ADD_EPA mean EPA_visit freq)
```

Fig. 14.2 Description of variables in the data file M_village

Results

variable name	storage type	display format	value label	variable label
ADD	int	%12.0g	ADDlabel	ADD identifier
ADD_EPA	float	%9.0g		Number of EPAs in ADD
ADD_vill	float	%9.0g		Number of villages in ADD from SPLU
ADD_hh	long	%12.0g		Number of HHs in ADD from SPLU
EPA	int	%14.0g	EPAlabel	EPA visited in GTIS
EPA_visit	float	%9.0g		Number of EPAs visited within ADD
EPA_vill	float	%9.0g		Number of villages in visited EPAs
EPA_hh	long	%12.0g		Number of HHs in visited EPAs from SPLU
village	int	%8.0g		village identifier
village_hh	int	%8.0g		Number of HHs in visited villages from SPLU
GTIS_hh	int	%8.0g		Number of HHs in visited villages from GTIS
hh_intv	int	%8.0g		Number of HHs interviewed in GTIS

Fig. 14.3 Number of EPAs in ADD and numbers visited

Results

```
. table ADD if village==1, contents(mean ADD_EPA mean EPA_visit freq)
```

ADD identifier	mean(ADD_EPA)	mean(EPA_vi~t)	Freq.
Blantyre	27	4	5
Karonga	9	3	3
Kasungu	26	4	4
Lilongwe	32	5	5
Machinga	33	4	4
Mzuzu	33	3	3
Salima	14	3	3
Shire Valley	11	1	3

There are two points to note with respect to results shown in ***Fig. 14.3***.

(a) The last two columns differ because there are missing values for one EPA in Blantyre ADD, and two EPAs in Shire Valley ADD. So in total, only 54 villages were enumerated although the original sampling scheme expected 60.

(b) Once the number of households in each selected EPA in the ADD has been determined, the results have to be scaled to ADD level. For each ADD, this will be done by taking the average number of households per EPA (using results from the selected EPAs) and scaling the result by the total number of EPAs in the ADD.

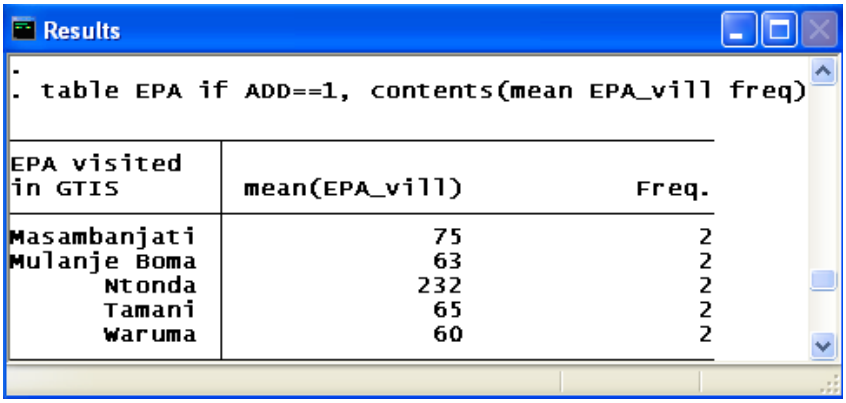
14.2 Scaling-up results from village to EPA and EPA to ADD

We consider here how the numbers of households enumerated in each of the two selected villages per EPA can be scaled to that EPA. The following command is used to illustrate the process, restricting attention to Blantyre ADD.

. table EPA if ADD==1, contents(mean EPA_vill freq)

The resulting table (see ***Fig. 14.4***) shows the number of villages in each of the five EPAs in Blantyre ADD, and the number of villages (variable **Freq**) selected from each EPA.

Fig. 14.4 Number of villages in EPAs of Blantyre and numbers selected



Browsing the data in `M_village.dta` shows that, in the EPA named Masambanjati (first EPA sampled in Blantyre ADD), there are 400 households found by the GTIS in the first village sampled, and 297 households found in the second village (see variable **GTIS_hh**). The average number of households per village for this EPA is therefore (400+297)/2 = 348.5.

Since this EPA has 75 villages (see ***Fig. 14.4***), the total number of households in this EPA may be estimated as being 348.5 * 75 = 26137.5.

Similarly, for the remaining EPAs in this ADD (apart from Ntonda for whom results were missing), the average numbers of households are 106 (for EPA Mulanje Boma), 94.5 (for Tamani) and 216 (for EPA Waruma). Hence the number of households in these 3 EPAs can be estimated by multiplying each of these estimates in turn by the number of villages in that EPA (from ***Fig. 14.4***) to give values 6678.0, 6142.5 and 12960.

The average number of households per EPA for Blantyre ADD can now be calculated as:

(26137.5 + 6678.0 + 6142.5 + 12960)/4 = 51918/4 = 12979.5.

But there are 27 EPAs in this ADD (see ***Fig 14.3***), and we have results only for 4 of these EPAs. Hence the total number of households in Blantyre ADD can be estimated as:

(51918 / 4) * 27 = 350446.5

A procedure similar to the above, gives the total number of households in the 8 ADDs as shown below, which adds up to 2,020,041 households in rural Malawi.

Blantyre	= 350447	Machinga	= 239382
Karonga	= 77172	Mzuzu	= 295730
Kasungu	= 177856	Salima	= 330997
Lilongwe	= 390058	Shire Valley	= 158400

14.3 Calculating the sampling weights

We have shown in the simple example above, how a population total can be determined using a straightforward scaling up procedure. This involved multiplying values in the data variable named **GTIS_hh** by certain scaling up factors, in a way that allowed data from village level to be scaled to EPA level, and then scaling up the EPA results to the ADD (strata) level. The steps involved were the following:

Step 1. Average the number of households in each pair of villages (within one EPA), i.e. dividing the variable **GTIS_hh** by 2.

Step 2. Scale up the average figures from above to EPA level by multiplying these figures by variable **EPA_vill.**

Step 3. Scale up the EPA level figures to ADD level by taking the average across EPA (i.e. dividing results from step 2 by the number of EPAs in ADD for which data are available – variable **EPA_visit**), and then multiplying the result by variable **ADD_EPA**, i.e. the number of EPAs in the ADD.

The Stata commands for this process are:

. egen villhhmn=sum(GTIS_hh/2), by(EPA) ← This is step 1

. recode villhhmn (0=.) ← Missing village values recoded back to missing after averaging

. generate EPAhh=villhhmn*EPA_vill ← This is step 2

. egen ADDhhmn=mean(EPAhh*ADD_EPA), by(ADD) ← Scaling up to ADD level. Taking the mean is equivalent to multiplying by variable EPA_visit. This is step 3.

. table ADD, contents(mean ADDhhmn) replace name(stat)

. table ADD, contents(sum stat1) row format(%9.0f)

Results from the last two statements above can be seen in ***Fig. 14.5***. It is seen that the above commands give an estimate of the rural population size as 2.02 million households. This is a more reasonable estimate than that produced by SPLU, compared to the 1998 census figure of 1.95 million.

Important: You will have observed that the datafile **M_village.dta** has been replaced by a new data file. Recall the previous data file by using the menu sequence **File, Open…** to obtain data in **M_village.dta**. Alternatively use:

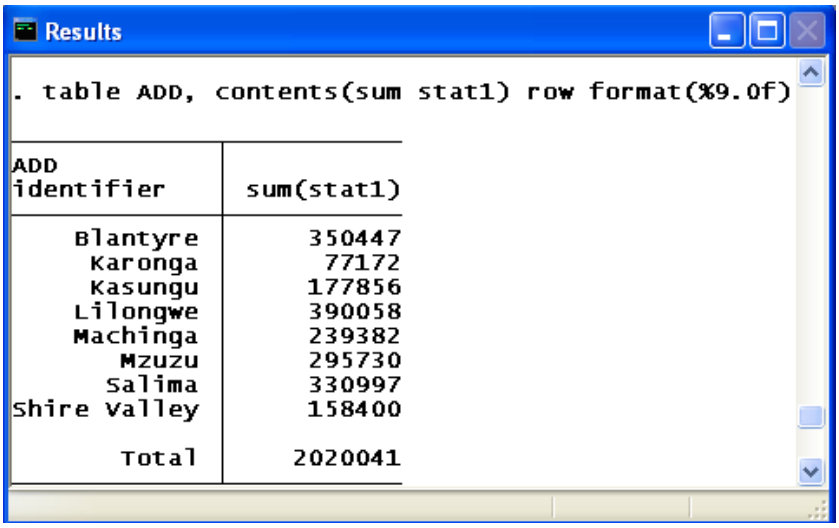
. use M_village, clear

We also note from the Stata commands above that the overall scaling up factor for each village (to ADD level) is computed by:

. generate w_total = (EPA_vill/2) * (ADD_EPA/EPA_visit)

The variable **w**_total is called the ***sampling weight***.

Fig. 14.5 Estimated number of households in each ADD using scaling up calculations



14.4 Estimating population totals

Although the process of calculating sampling weights for a simple sampling structure was explained in several steps in the previous section, in practice it would only be necessary to compute a variable (**w_total** above) to hold the sampling weights appropriate for the sampling scheme used. This variable denotes the inverse of the probability that the observation is included due to the sampling design.

Once the sampling weights have been computed for each sampling unit, estimating the population total is quite straightforward. The Stata command for this is:

. table ADD [pweight=w_total], contents(sum GTIS_hh) row format(%9.0f)

The results from this are shown in ***Fig. 14.6***. This is identical to results produced in ***Fig. 14.5***. The effect of the ***pweight*** option has been to multiply each row of **GTIS_hh** by **w_total** prior to summing up.

We can also use the SPLU data on the number of households in the selected villages to generate an estimate of the population total using the same sampling weights. The command below produces the results shown in ***Fig. 14.7***. The population total is here estimated as being approximately 2.6 million households.

. table ADD [pweight=w_total], contents(sum village_hh) row format(%9.0f)

Fig. 14.6 Estimated number of households in each ADD from GTIS survey

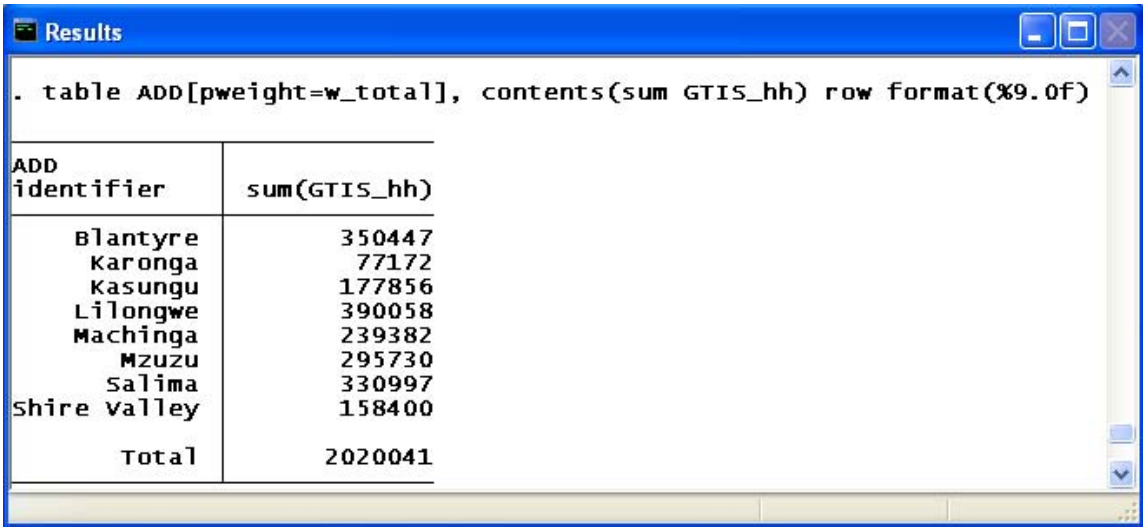
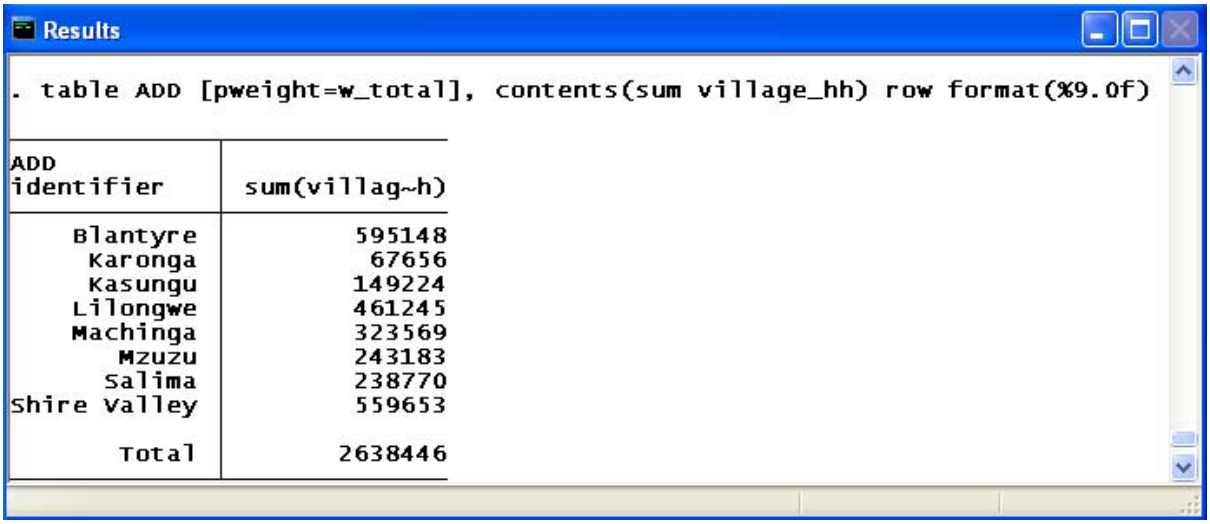


Fig. 14.7 Number of households in each ADD using SPLU registrations



To get just the grand total only, the **tabstat** command has to be used as follows:

tabstat GTIS_hh[aweight=w_total], statistics(sum)

tabstat village_hh[aweight=w_total], statistics(sum)

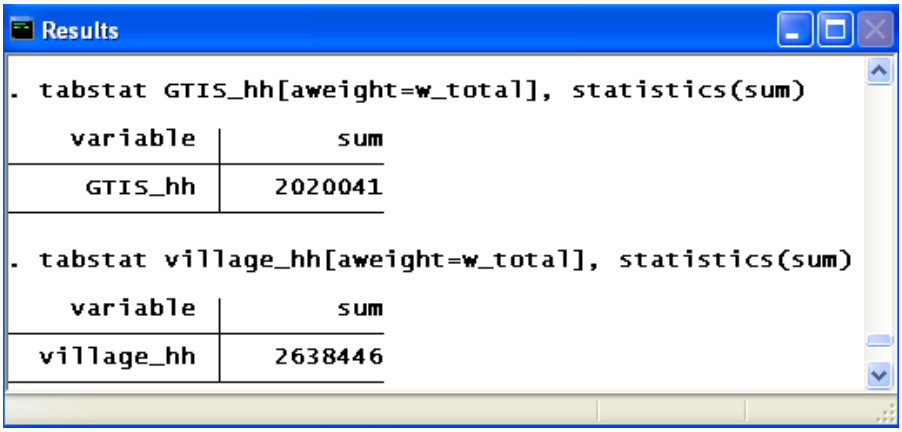
The results are shown in ***Fig. 14.8.***

The **tabstat** command does not have the option *pweight*, but the option *aweight* can be used instead. Stata defines analytical weights as "inversely proportional to the variance of an observation". The very specific form of design used here gives the same results as with probability weights. Hence a table with the same figures as those in ***Fig. 14.6*** can be obtained using:

. tabstat GTIS_hh[aweight=w_total], statistics(sum) by(ADD) format(%9.0f)

Try this and see whether it works!!

Fig. 14.8 Overall numbers of households in Malawi from GTIS and SPLU data



14.5 A self-weighting sampling scheme

The GTIS adopted a simple random sampling scheme for its two stages and decided to sample 3 to 5 EPAs per ADD, and 2 villages per EPA.

Let us now suppose that the total number of villages in all EPAs was known and that, within each ADD, the EPAs were chosen with probability proportional to size (PPS) sampling. Here "size" of the EPA will be taken as the number of villages in the EPA. We will keep the next stage of sampling the same, i.e. selecting 2 villages with simple random sampling.

Let us also suppose that the above sampling procedure led to the same EPAs available in the dataset **`M_village.dta`**.

Considering the data shown in ***Fig. 14.9***, we can then see that the probability of selecting the first EPA in Blantyre ADD is 75/3119. The inverse of this probability gives the contribution to the sampling weight from this EPA. However, since PPS sampling is essentially sampling with replacement, each EPA will give an independent estimate for the number of households in Blantyre ADD by using, in turn, sampling weights of 3119/75, 3119/63, 3119/232, 3119/65 and 3119/60. The average of these estimates (i.e. using as divisor **`EPA_visit`**) will give the overall estimate for the total number of households in Blantyre. Hence the correct sampling weight for each EPA is computed as: **ADD_vill/(EPA_vill*EPA_visit)**.

Fig. 14.9 Number of villages in selected EPAs and in Blantyre as a whole

Results

```
. list ADD ADD_vill EPA_vill in 1/10

     +-------------------------------+
     |      ADD   ADD_vill   EPA_vill |
     |-------------------------------|
  1. | Blantyre       3119         75 |
  2. | Blantyre       3119         75 |
  3. | Blantyre       3119         63 |
  4. | Blantyre       3119         63 |
  5. | Blantyre       3119        232 |
     |-------------------------------|
  6. | Blantyre       3119        232 |
  7. | Blantyre       3119         65 |
  8. | Blantyre       3119         65 |
  9. | Blantyre       3119         60 |
 10. | Blantyre       3119         60 |
     +-------------------------------+
```

At the next stage of sampling, the probability of selecting 2 villages from all villages in a selected EPA = **2/EPA_vill**. Hence the contribution to the sampling weight here is given by **(EPA_vill/2)**.

Hence the overall sampling weight for a village in one ADD can be computed as:

(EPA_vill/2) * (ADD_vill/(EPA_vill*EPA_visit) = ADD_vill/(2*EPA_visit)

The result above is a constant within any given ADD. Hence any village in a given ADD has the same sampling weight, i.e. each village has the same chance of selection. Such a sampling scheme is called a **self-weighting design**, the weights being the inverse of the probabilities of selection. These weights can be calculated using:

. generate sw_total= ADD_vill/(2*EPA_visit)

Once the sampling weights have been generated, the total number of rural households in each ADD, and in the whole of Malawi can be obtained from:

. table ADD [pweight=sw_total], contents(sum GTIS_hh) row format(%9.0f)

The results of this can be seen in ***Fig. 14.10***.

The advantage of the self-weighting scheme is that the mean number of households per village can be computed for any given ADD as the simple average of the household numbers from the sampled villages. So for example, the simple average of the number of households per village in Blantyre ADD = (400+297+172+40+125+64+280+152)/8 = 191.25. This result multiplied by the total number of villages in all Blantyre, i.e. 3119, gives and estimate of the total number of households in Blantyre as 596,509. This coincides with the result for Blantyre shown in ***Fig. 14.10*** above.

See ***Fig. 14.11*** for results from other ADDs. Taking the product of the two numerical columns in this figure will give the results of ***Fig. 14.10***.

Fig. 14.10 Estimated number of households per ADD assuming a self-weigting design

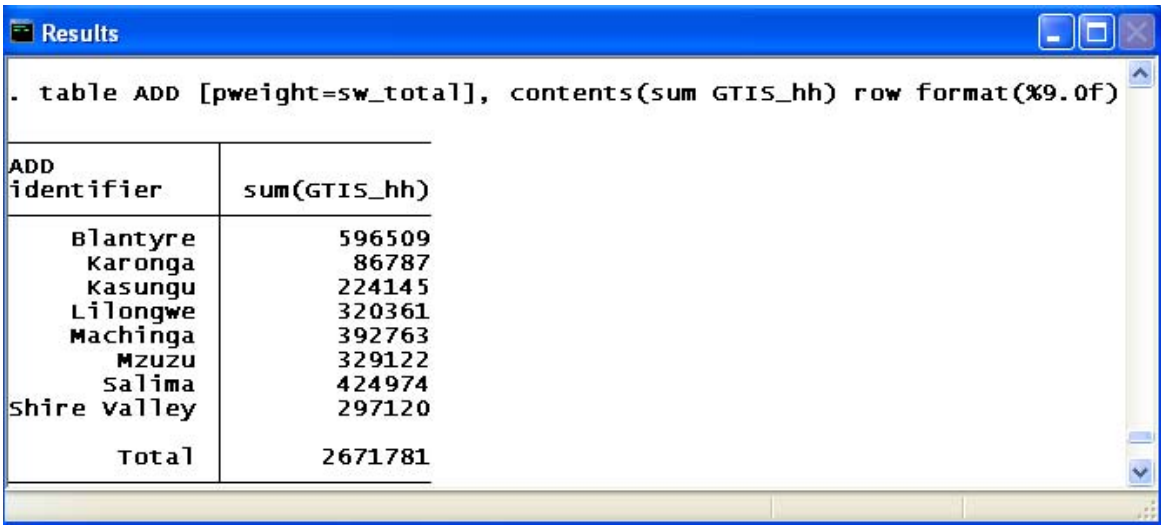
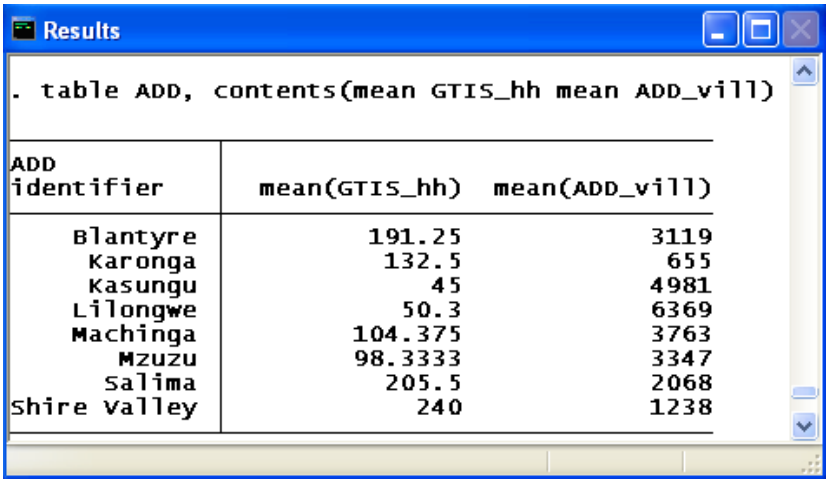


Fig. 14.11 Summaries of household numbers per ADD



14.6 Keeping a record of the analysis

When analyses are weighted it may be harder for other staff and particularly readers, to check the results. This is risky. Also if further analyses give different results it may not be clear whether this is due to differences in the data or in the way the weighting was done.

In chapter 5 we stressed the importance of using DO files to record an audit trail of the analyses. The same applies here, to clarify exactly how a weighted analysis was conducted. We therefore provide the DO file named

Chapter 14 weights ***.do*** where the analyses conducted in this chapter were recorded.

Chapter 15 Standard errors for totals and proportions

In Chapter 14 we showed how sampling weights can be used to derive an estimate of the population total for the Malawi Ground Truth Investigation Study (GTIS) survey. We got a point estimate of 2,020,041 for the total number of rural households in Malawi. It is also important to quantify the precision of our estimate, by deriving its standard error.

In this chapter we use Stata to compute standard errors of means and totals. We also give confidence intervals for the true value of population parameters. We do this in two contexts: first we assume that the sampling scheme of the GTIS is simple random, then we take into account the stratified multistage sampling scheme of the GTIS.

In Chapter 14 we saw that the GTIS survey data from 60 villages is available in the datafile **`M_village.dta`**. Recall that 6 villages could not be located, so only 54 villages provide information. This may be checked with:

. use M_village, clear

. desc

. list GTIS_hh if GTIS_hh==.

15.1 Motivation of the standard error of the sample mean

Assume we were asked: what is the true average number of households per village in Malawi from the GTIS?

To reflect the sampling variability of our estimate, we can quantify its precision with its standard error, which is a function of the variability in the number of households in the sampled villages and the number of villages sampled. The greater the number of villages surveyed, the smaller is the standard error, so the more precise our estimate will be and the narrower our confidence interval.

Initially we ignore the stratification of the survey design and assume that the 54 villages were drawn as a simple random sample of the 25,540 villages in Malawi: standard theory sets the standard error of the sample mean to $s/\sqrt{n}$, where s=sample standard deviation and n=number of villages sampled, assuming that the sample size n is very small relative to the size of the population.

We can combine our estimate with its measure of precision to give a range which is highly likely to contain the true value of the population parameter. This range is known as a confidence interval. Conventionally 95% confidence limits are calculated, i.e. we can be 95% certain that the confidence limits include the true population parameter.

Assuming that the sample mean is normally distributed, the 95% confidence limits are approximately given by

sample mean $\pm$ t * standard error of the sample mean

The t multiplier comes from the t-distribution with degrees of freedom (d.f.) equal to the number of villages sampled –1, i.e. 53. When the sample size is large (say > 50 d.f.), the t-multiplier is about 2.

This is the default method inbuilt in the Stata `ci` command. Try it with:

. ci GTIS_hh

This gives a mean of 117.148 households per village with a standard error of 13.04 households, and a 95% confidence interval for the true population mean of households per village of 91 to 143 households.

15.2 The finite population correction factor (fpc)

The method used above assumes that the villages are sampled from an infinite number of villages. But this is not the case, because we know the total number of villages in Malawi to be 25,540 villages, and we also have a sampling frame with the total number of registered households from the SPLU.

Hence we know the proportion of sampled villages, i.e. 54/25,540=0.00211. The way to include this knowledge into the estimation process is to multiply the standard error of the mean, as produced by Stata above, by √(1-f), where f is n/N, or the proportion of units sampled from the population of all units. The factor (1-f) is known in survey work as the finite population correction or FPC.

Now check what summary data is temporarily stored in Stata with:

. return list

this shows that the standard error has been saved in the scalar named r(se), so use

. display (sqrt(1-54/25540))*r(se)

to get a revised estimate of the standard error as 13.026. This is not much of a change from the previous value of 13.04, because the finite population correction factor is 0.9979 since we sampled a tiny proportion of all households in Malawi. Nevertheless, this illustrates the principle that the larger the proportion of sampled units in the population, the more precise our estimate will be. In theory, if a census provided complete coverage of a population, a standard error will not be necessary since there is no sampling variability.

The 95% confidence interval for the true value of the population mean could now be recalculated with the more correct standard error of 13.026, although here it makes hardly any difference, especially after rounding to integer numbers.

15.3 The standard error of the sample total

To estimate the average number of households per village is purely a technical exercise, because there is no such thing as an “average” village. Recall from Chapter 14 that a primary objective of the GTIS was to estimate the total number of rural households in Malawi. If we assume that the 54 villages have been sampled at random from the population of all villages, then the total number of households in Malawi can be estimated by multiplying the mean we obtained earlier, i.e. 117.148, by the number of villages, i.e. 25,540, to give the result 2,991,960 households.

How do we now get a standard error for this estimated total? First recognise that the total was calculated by using the result that $T=\Sigma x=N\bar{x}$, where N=25,540 and $\bar{x}$ = 117.148.

The following commands will display the answer:

. tabstat GTIS_hh, stat(mean count)

. display 25540*117.148

We already know the standard error of the mean, i.e. s.e($\bar{x}$). We know N is a fixed quantity as it does not vary, i.e. it has no standard error. Hence:

s.e.(T) = N * s.e($\bar{x}$) = 25540 * s.e($\bar{x}$)= 25540 * 13.04 = 333042

Thus the estimate of the total number of households in Malawi is 2,991,960 households with a standard error of 333,042.

We could now use the standard method to compute a 95% confidence interval for the true total number of households in Malawi, as

estimate(country total) ± (t on 53 d.f.) * s.e.(country total).

However, it is simpler to multiply by the factor N the results of the 95% CI for the mean stored by Stata as seen by the first two commands below.

. ci GTIS_hh

. return list /*to see what Stata temporarily stores*/

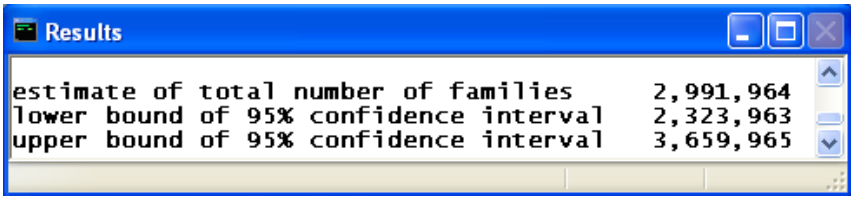
display _newline "estimate of total number of households " %12.0fc 25540*r(mean), ///

_newline "lower bound of 95% confidence interval" %12.0fc 25540*r(lb), ///

_newline "upper bound of 95% confidence interval" %12.0fc 25540*r(ub)

The corresponding output from the last 3 statements above is shown in ***Fig. 15.1***.

Fig. 15.1 Estimate of total number of households with 95% confidence limits



Thus this simple method, which considers the surveyed villages as a simple random sample and ignores the survey design, gives the range 2,323,963 to 3,659,965 as a 95% confidence interval for the true total number of households in Malawi from the GTIS survey.

A confidence interval of plus or minus over half a million households should not be surprising, because such wide confidence intervals are common when estimating totals.

15.4 Using Stata's special commands for survey work

Stata has powerful features to deal with estimation in the context of survey work, starting with the command ***svyset***, which is used to specify the survey design. So, we can quickly reproduce the calculations in section 3 by setting the same weight for all 54 villages. The probability of selecting a village, when using simple random sampling, is 54/25540. Hence a weight variable for the analysis can be generated using:

. gen weight = 25540/54 /* this is the inverse of the probability of selection */

and specifying the survey design as a simple random sample (option srs) with:

. svyset _n [pweight=weight], clear

The clear option removes any pre-existing design specifications.

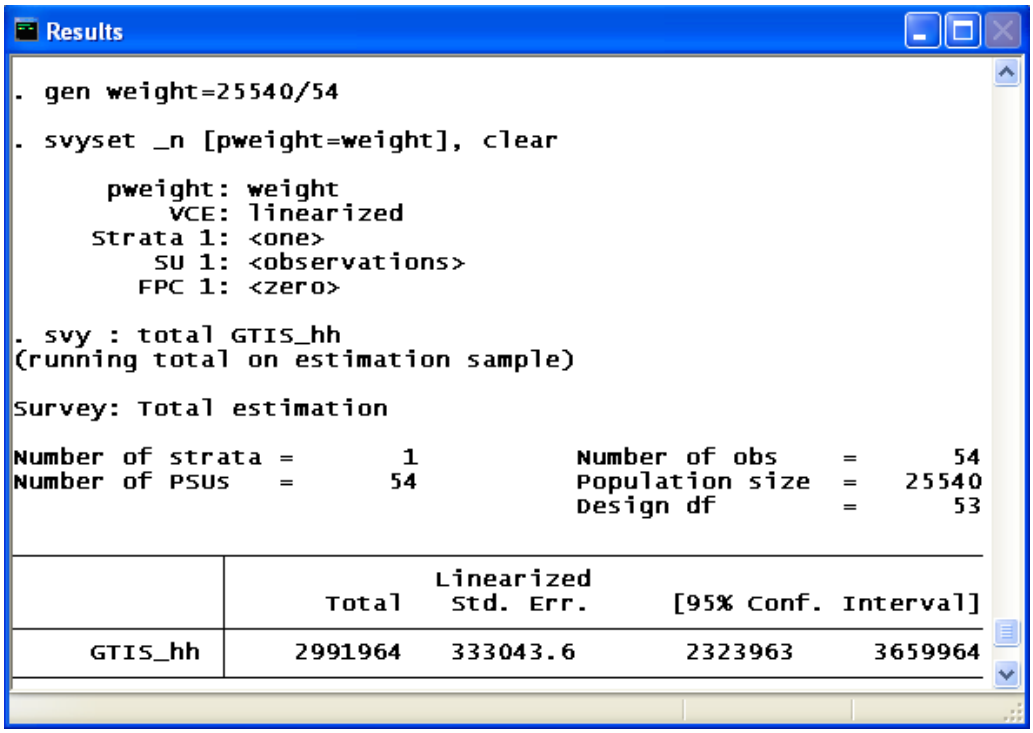
Finally estimate the total number of households and its a 95% confidence interval with:

. svy : total GTIS_hh

whose output is shown in ***Fig. 15.2***.

Observe that all estimates in the table output in ***Fig. 15.2*** are the same as those derived from first principles in ***Fig. 15.1***. This is because when using the ***svyset*** command we specified the survey design as a simple random sample.

Fig. 15.2 Results for total number of households using svy commands



15.5 Considering the survey design

As seen in chapter 14, ignoring the stratification and clustering of the survey design is not the most the most efficient use of the available information. We saw that by taking into account the survey design we were able to compute different probabilities of selection for selecting EPAs within ADDs, and for selecting villages within EPAs. We then used these to derive sampling weights (in a variable called **w_total**), which were different for each sampled EPA.

Now check if the data file **`M_village.dta`** that you have opened includes the variable **w_total**. If not, generate this (as was done in Chapter 14) using:

. generate w_total = (EPA_vill/2) * (ADD_EPA/EPA_visit)

Note that there is a clear motivation for considering the 3 elements of a survey design (sampling weights, clustering and stratification). In the Stata User manual page 345: the effects of these are described as follows:

1. Including sampling weights in the analysis gives estimators that are much less biased than otherwise. Sampling weights are described as the inverse of the probability of a unit to be sampled. However, post-sampling adjustments are often done using weights that correspond to the number of elements in the population that a sampled unit represents.
2. Villages were not sampled independently but from within each selected EPA. Ignoring this non-independence within clusters (EPAs) results in artificially small standard errors.
3. Sampling of clusters is done independently across all strata, whose definition is determined in advance. In the Malawi GTIS study, the strata were represented by the 8 ADD into which the country is divided. Usually, samples are drawn from all strata and, because of the independence across strata, this produces smaller standard errors.

Basically, we use sampling weights to get the right point estimate, and we consider clustering and stratification to get the right standard errors.

When the survey design was taken into consideration, we saw the estimate of the total number of households to be about 2.02 million households (Section 14.4). We would expect the standard error of this estimate to be different from that obtained assuming simple random sampling - generally it should increase. For details of the methodology for computing standard errors of complex survey designs, see the Survey Data Reference Manual.

In the Malawi GTIS survey, the design was a stratified 2-stage cluster sample with ADD as strata, EPA as primary sampling units (clusters). The sampling weights could be specified using the dialog box from the menu selection **Statistics ⇒ Survey Data Analysis ⇒ Setup & Utilities ⇒ Declare survey design for data set,** and filled as shown in ***Fig. 15.3***, followed by the dialogue under the **Weights** option as shown in ***Fig. 15.4***.

Fig. 15.3 Main dialogue for setting design

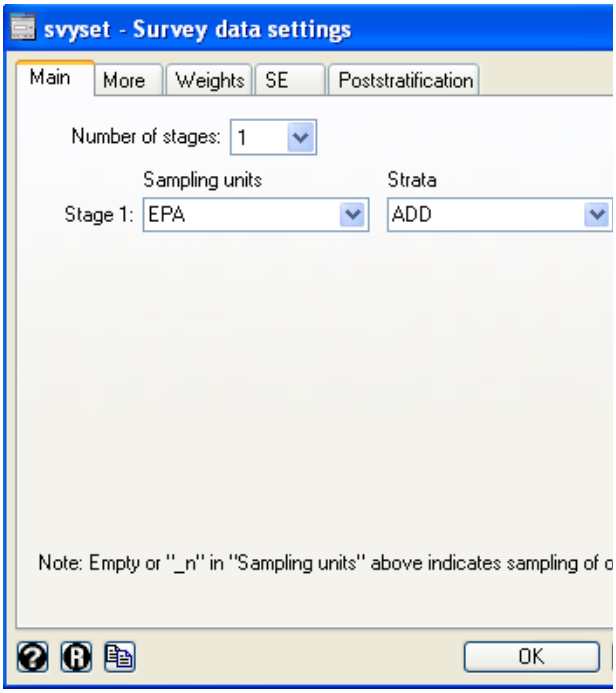
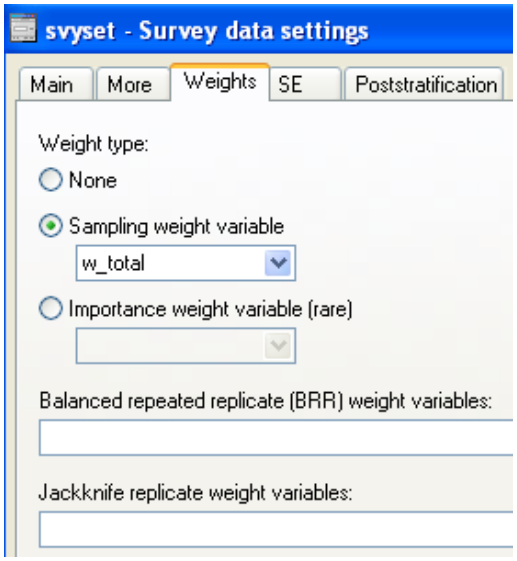


Fig. 15.4 Dialogue for Weights option



The 'clear' option deletes any pre-existing specifications of the survey design. Clicking **OK** produces the commands shown below

. svyset EPA [pweight=w_total], strata(ADD) vce(linearized)

Note that we are not providing information about the secondary sampling units, i.e. the villages, because Stata uses methods for computing standard errors at the PSU level only.

We can now revise the estimate of the total and its 95% confidence interval with:

. svy : total GTIS_hh

This produces the output shown in ***Fig. 15.5*** which gives the same estimate as in ***Fig. 14.6*** (of Chapter 14), but a standard error is not computed. This is because Stata detects, via the ***svyset*** command, that ADD 8 has only a single PSU, which is EPA number 29.

We can get a standard error by omitting ADD 8, with the following command, but this would give an underestimate of the overall population total of household numbers in rural Malawi (see ***Fig. 15.6***).

. svy : total GTIS_hh if ADD!=8

Fig. 15.5 Results using all ADDs

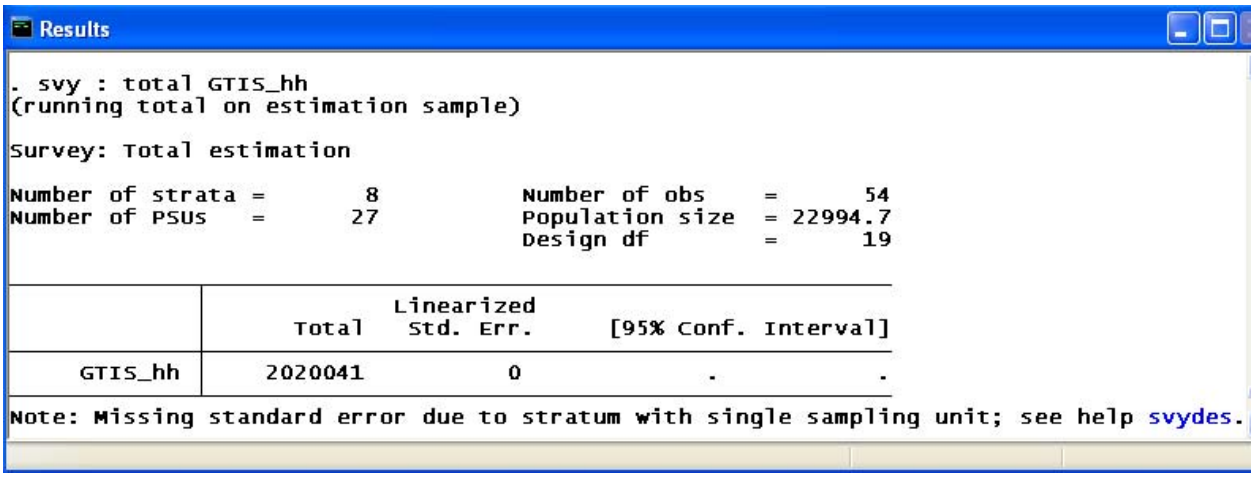
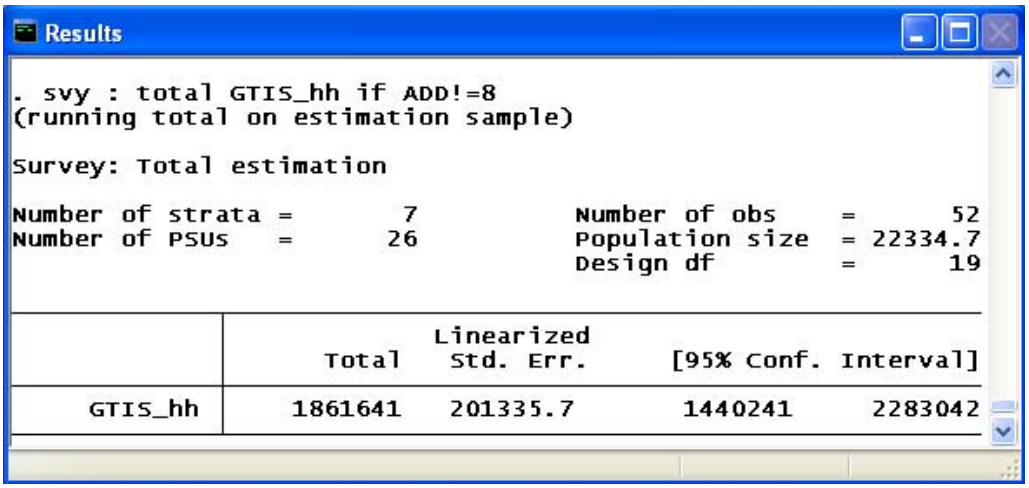


Fig. 15.6 Results omitting ADD 8 which has only one EPA



The point estimate of 1,861,641 households in ***Fig. 15.4*** is different from that shown in ***Fig. 14.6*** because here one ADD has been omitted. The difference corresponds to the (rather imprecise) estimate shown for Shire Valley in ***Fig. 14.6***.

The ***svy*** commands of Stata can also compute standard errors and confidence intervals for each stratum separately. Try this with:

. svy : total GTIS_hh if ADD! = 8) , over(ADD)

The output is shown in ***Fig. 15.7***. Notice that the ADD estimates coincide with figures shown in the previous chapter in ***Fig. 14.6*** (apart from the missing ADD). The benefit of using ***svy*** commands is the inclusion of standard errors for the ADD estimates of the size of Malawi's rural population.

The loss in precision of a complex survey design can be measured by a quantity called **Deff**, which stands for the ***design effect***. **Deff** is the ratio of the design-based variance estimate to the estimate of variance from treating the survey design as a simple random sample. Here we have specified the survey design as taking place with EPAs chosen at random, and then villages chosen at random. To obtain the deff estimates use:

. estat effects

The results can be seen in ***Fig. 15.8***.

Fig. 15.7 Estimates by ADD of total numbers of households with 95% confidence limits

Results

```
. svy : total GTIS_hh if ADD! = 8 , over(ADD)
(running total on estimation sample)

Survey: Total estimation

Number of strata =       7          Number of obs     =       52
Number of PSUs   =      26          Population size   = 22334.7
                                    Design df         =       19

     Blantyre: ADD = Blantyre
      Karonga: ADD = Karonga
      Kasungu: ADD = Kasungu
     Lilongwe: ADD = Lilongwe
     Machinga: ADD = Machinga
        Mzuzu: ADD = Mzuzu
       Salima: ADD = Salima
```

Over	Total	Linearized Std. Err.	[95% Conf.	Interval]
GTIS_hh				
Blantyre	350446.5	125578.3	87608.18	613284.8
Karonga	77172	14089.09	47683.2	106660.8
Kasungu	177856.3	20647.65	134640.2	221072.3
Lilongwe	390057.6	65016.28	253977	526138.2
Machinga	239382	111708.9	5572.506	473191.5
Mzuzu	295729.5	68280.05	152817.7	438641.3
Salima	330997.3	52660.79	220777	441217.6

Fig. 15.8 Estimates by ADD with associated design effects

Results

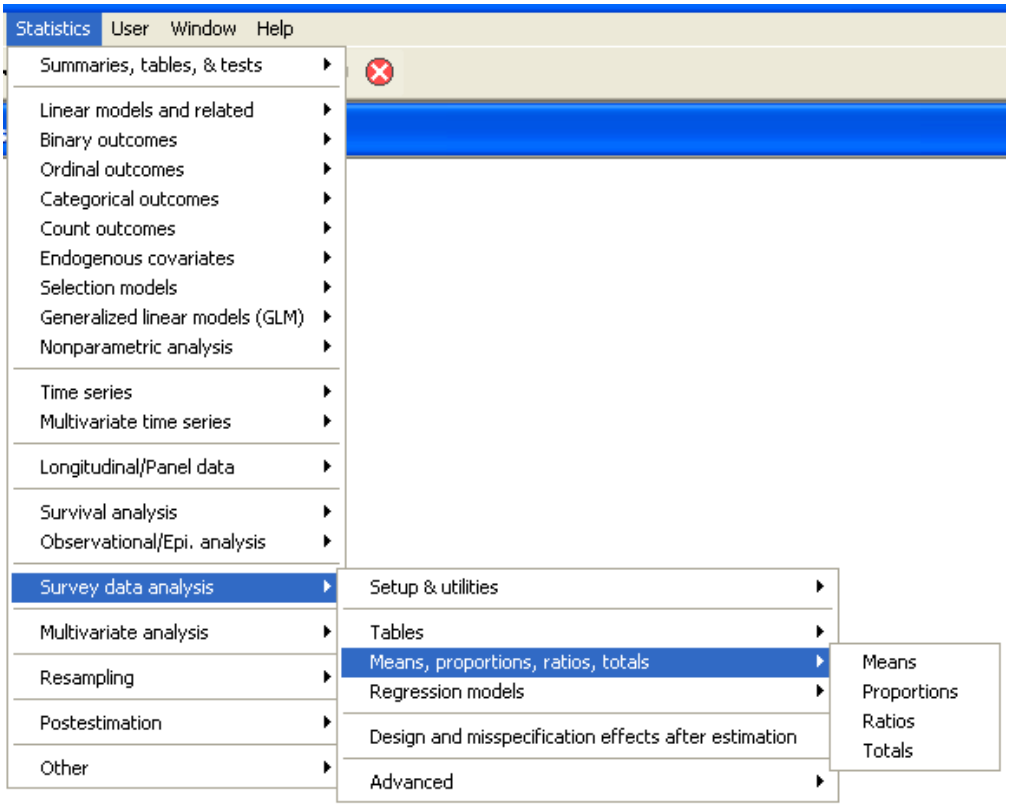
```
.
. estat effects

     Blantyre: ADD = Blantyre
      Karonga: ADD = Karonga
      Kasungu: ADD = Kasungu
     Lilongwe: ADD = Lilongwe
     Machinga: ADD = Machinga
        Mzuzu: ADD = Mzuzu
       Salima: ADD = Salima
```

Over	Total	Linearized Std. Err.	Deff	Deft
GTIS_hh				
Blantyre	350446.5	125578.3	.405329	.636654
Karonga	77172	14089.09	.04117	.202903
Kasungu	177856.3	20647.65	.129864	.360366
Lilongwe	390057.6	65016.28	.574317	.757837
Machinga	239382	111708.9	.673113	.820435
Mzuzu	295729.5	68280.05	.398646	.631384
Salima	330997.3	52660.79	.091364	.302265

15.6 Standard errors for other sample estimates

Stata can also estimate standard errors from complex survey designs for other non-model based estimates like means, proportions and ratios. The respective commands are ***svy:mean***, ***svy:prop*** and ***svy:ratio***. Dialog boxes for these estimators are accessible from the selection **Statistics ⇒ Survey Data Analysis ⇒ Means, proportion, ratios, totals** in the main menu, as shown in ***Fig. 15.9***.

Fig. 15.9 Dialogue for getting estimates and standard errors of other characteristics



15.7 Standard errors for proportions

It was of interest to estimate the proportion of households that had registered for the Starter Pack distribution in the 1999/2000 season, out of all those from the GTIS village mapping. Information is available on how many members of each household had registered. Households were then grouped into 3 categories depending on how many members had registered: zero, one, two or more. This is because officially only one member per household could register.

Data at the household level are stored in the file **`M_household.dta`**.

Open the dataset and look at its variables with:

. use M_household, clear
. describe

We start by simply tabulating the information, on numbers of registered members, with:

. tab register

The results are shown in ***Fig. 15.10.*** It may be observed that in the villages mapped by the GTIS, about 7% of households had not registered, about 65% had registered correctly, and about 28% of households had multiple members registered.

Fig. 15.10 Frequency distribution for numbers registered per household

```
Results

. tab register

  household |
    members |
 registered |
  for SP in |
      99/00 |      Freq.     Percent        Cum.
------------+-----------------------------------
       zero |        446        7.34        7.34
        one |      3,929       64.67       72.02
       two+ |      1,700       27.98      100.00
------------+-----------------------------------
      Total |      6,075      100.00
```

The percentages given in ***Fig. 15.10*** assume that the sample of 54 villages was drawn completely at random. But we know that this is not the case, so let's re-use the sampling weights computed earlier and define the stratified multistage sampling scheme adopted by the GTIS with:

. svyset EPA [pweight=w_total], clear strata(ADD)

Then, to incorporate this information into the estimation process use:

. svy : prop register if ADD!=8   */recall ADD 8 only had 1 EPA */

The output from ***svy:prop*** is shown in ***Fig. 15.11***.

Fig. 15.11 Proportions registered allowing for sampling weights

```
Results

. svyset EPA [pweight=w_total], clear strata(ADD)

      pweight: w_total
          VCE: linearized
     Strata 1: ADD
         SU 1: EPA
        FPC 1: <zero>

. svy : prop register if ADD!=8
(running proportion on estimation sample)

Survey: Proportion estimation

Number of strata =       7          Number of obs     =      5603
Number of PSUs   =      26          Population size   =  1.8e+06
                                    Design df         =        19

      _prop_3: register = two+

--------------------------------------------------------------
             |             Linearized       Binomial Wald
             |  Proportion   Std. Err.     [95% Conf. Interval]
-------------+------------------------------------------------
register     |
        zero |    .0725626   .0094344     .0528162     .092309
         one |    .5985631   .0304663     .5347965    .6623298
     _prop_3 |    .3288743   .0252943     .2759327    .3818159
--------------------------------------------------------------
```

15.8 The use of the svy:tab command

The normal approximation works well with this large dataset. However, with smaller datasets and fewer observations, the standard errors will be larger. Then, if the point estimate of the proportion is very close to 0 or 1, the normal approximation may well give confidence limits outside the range 0 to 1.

Fortunately, Stata provides the ***svy:tab*** command, which computes confidence intervals more exactly on the basis of the Binomial distribution. This ensures the confidence bounds are always between 0 and 1. The following commands provide an illustration.

. set matsize 100 /* needed for the display */

. svy: tab ADD register if ADD!=8, row se ci format(%4.1f) percent

As shown in ***Fig. 15.12***, the output from ***svy:tab*** can be customised to make it more readable for presentation purposes. This command does not however present just the margins, i.e. the last row entry named "Total", without a breakdown by ADD.

Fig. 15.12 Estimated proportions assuming a binomial distribution

```
. svy : tab ADD register if ADD!=8, row se ci format(%4.1f) percent
(running tabulate on estimation sample)

Number of strata   =         7                  Number of obs      =       5603
Number of PSUs     =        26                  Population size    = 1783612.8
                                                Design df          =         19

------------------------------------------------------------------------------
name of |    household members registered for SP in 99/00
ADD     |       zero          one          two+        Total
--------+---------------------------------------------------------------------
Blantyre|        7.9         59.2         32.9        100.0
        |      (2.9)       (13.5)       (10.7)
        |  [3.6,16.5]  [31.1,82.4]  [15.2,57.4]

 Karonga|        9.5         70.9         19.6        100.0
        |      (0.9)       (5.2)        (5.6)
        |  [7.9,11.6]  [58.9,80.6]  [10.4,33.8]

 Kasungu|        9.4         55.2         35.3        100.0
        |      (2.2)       (4.2)        (4.4)
        |  [5.7,15.3]  [46.4,63.8]  [26.7,45.0]

Lilongwe|        6.5         54.8         38.7        100.0
        |      (2.2)       (3.0)        (2.9)
        |  [3.2,12.7]  [48.5,61.0]  [32.9,44.8]

Machinga|        2.7         60.1         37.2        100.0
        |      (0.6)       (6.4)        (7.0)
        |   [1.7,4.4]  [46.2,72.5]  [24.0,52.6]

   Mzuzu|        8.2         63.0         28.7        100.0
        |      (2.5)       (6.3)        (5.6)
        |  [4.3,15.3]  [49.2,75.0]  [18.5,41.8]

  Salima|        8.2         63.9         27.9        100.0
        |      (1.7)       (1.6)        (2.3)
        |  [5.3,12.4]  [60.6,67.1]  [23.3,33.1]

   Total|        7.3         59.9         32.9        100.0
        |      (0.9)       (3.0)        (2.5)
        |   [5.5,9.5]  [53.3,66.0]  [27.8,38.4]
------------------------------------------------------------------------------
  Key:  row percentages
        (linearized standard errors of row percentages)
        [95% confidence intervals for row percentages]

  Pearson:
    Uncorrected   chi2(12)         =   87.1449
    Design-based  F(3.10, 58.85)   =    0.7053     P = 0.5569
```

Chapter 16 Statistical modelling

This chapter proposes a systematic approach to statistical modelling, using a regression example. We use the data file from the rice survey **paddyrice.dta** described in Section 0.2.4. In this chapter we begin by ignoring the survey design, i.e. we assume that data was collected as a simple random sample. Then we extend these ideas to take into account the survey design.

16.1 A systematic approach to statistical modelling

One of the two objectives of the rice survey was to examine the relationship between rice yield and the different cultivation practices. If we ignore the field variable that is just a numeric identifier, there are four variables providing information about cultivation practices, as shown in ***Fig. 16.1***. These are: **village, size** (of field in acres), (bags of) **fertiliser** and **variety** (grown). To draw an analogy with designed experiments, **village** is the equivalent of blocks and **size** is the equivalent of a covariate. These cannot be modified, whereas **fertiliser** and **variety** are the equivalent of treatments and can be modified by the farmer to influence the rice yield.

Fig. 16.1 Browsing the paddyrice data

Data Browser

Preserve | Restore | Sort | << | >> | Hide | Delete...

yield[1] = 53.599998

	yield	village	field	size	fertiliser	variety
1	53.6	SABEY	3	2	2.5	OLD
2	44.6	SABEY	13	5	1.5	OLD
3	50.7	SABEY	4	5	3	OLD
4	33.6	SABEY	20	1.5	1	TRAD
5	62.1	SABEY	19	5	2.5	NEW
6	30.6	SABEY	10	4	1.5	TRAD
7	37.7	SABEY	8	4.5	2	OLD
8	24.3	SABEY	11	3.5	0	TRAD
9	56.8	SABEY	14	5	2	NEW
10	59.3	SABEY	12	4.5	2.5	OLD
11	40.4	KESEN	9	1.5	2	OLD
12	25.8	KESEN	8	7	0	TRAD

Moreover, just as important for statistical modelling work, is that **size** and **fertiliser** are numeric variables, whereas village and variety are categorical variables. This is obvious in ***Fig. 16.1*** where text has been stored for **village** and **variety**.

All four factors represent cultivation practices and could be assessed together for their influence on rice yield by including them all in the same statistical modelling process. However, for the sake of simplicity, here we only include two factors: fertiliser and variety.

When assessing only numerical variables, we can use:

. regress yield fertiliser

When assessing only categorical variables we can use:

. oneway yield variety

But to assess the influence on yield of both factors together, a linear model must be used instead. This is a generalisation that allows both numeric and categorical variables in the same model. In Stata we can do this using either the **anova** or **xi:regress** commands.

A systematic approach to statistical modelling is to go through 5 steps: exploratory stage, comparing competing models, fitting the chosen model, checking assumptions, presenting results. The data analysis below therefore starts with an exploratory stage using plots of the observed data that represent the structure in the data.

Note that the emphasis here is on exploring relationships as in Chapter 12, rather than on estimating ideas covered in Chapters 14 and 15.

16.2 Exploratory stage

The structure underlying the data are the levels of fertiliser and variety grown, so start with a scatter plot of the variable yield against fertiliser, split by each type of variety, using

. scatter yield fertiliser, by(variety, rows(1))

which creates a scatter plot split into 3 separate panels as shown in ***Fig. 16.2***.

Fig. 16.2 Graph of yield by fertiliser for each variety

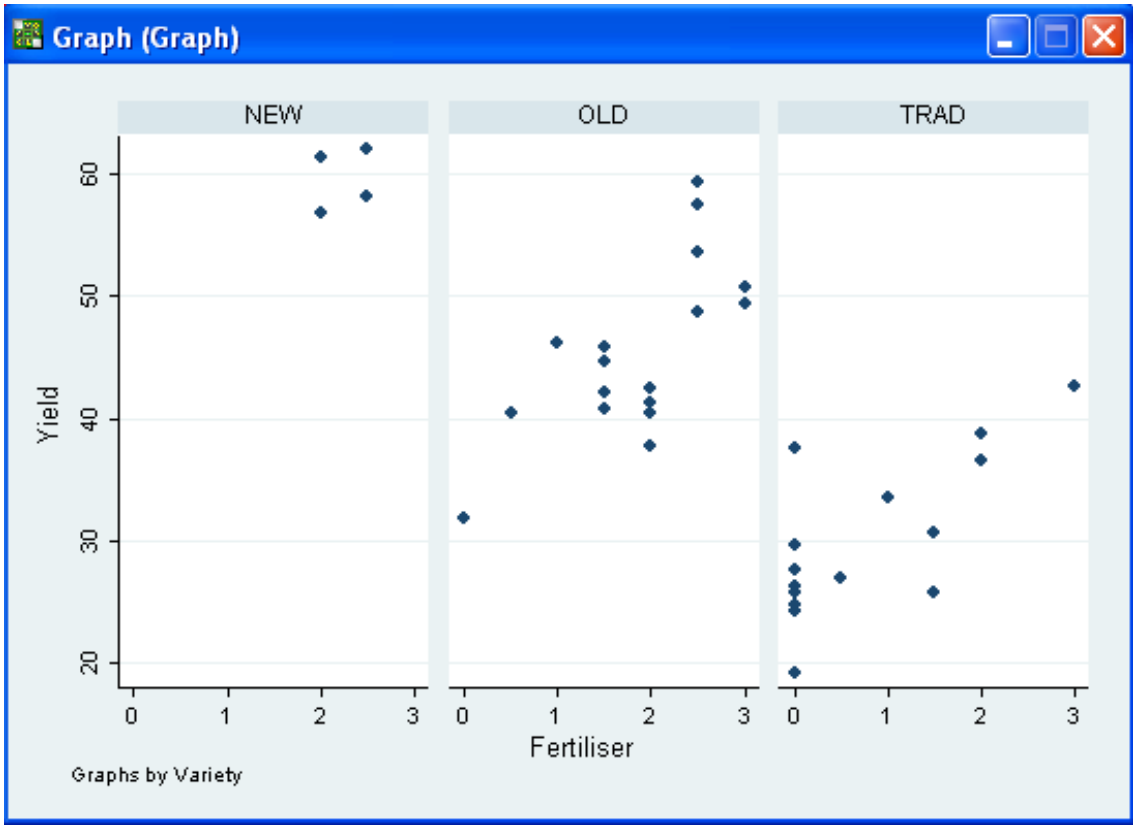


Fig. 16.2 is for exploratory purposes, so it does not need extra customising. It shows that all 3 varieties seem to have roughly the same response to fertiliser application but average yields are higher for the **NEW** variety and lowest for the **TRAD** variety. The change in yield for a unit increase in fertiliser amounts seems constant at all levels of fertiliser. Translated into a statistical model, this means the same slope (i.e. same rate of increase) and a different intercept (or constant) for each variety, i.e. a set of 3 parallel straight lines.

16.3 Model comparison and selection

We need a formal way of deciding if the 3 intercepts are different but the 3 slopes are not. We do this by comparing competing models using the linear model framework as:

Data = pattern + residual

Here pattern (structure) represents the **systematic** component of the model. It includes terms that may potentially explain variability in the data (the key response of interest). The **residual** includes the random, unexplained component of variability.

The description of the model as linear does not necessarily mean a straight line, but that the terms are included into the pattern one after the other, i.e. terms are additive. In the pattern part of the statistical model here we consider 2 terms for inclusion: fertiliser and variety. In increasing level of complexity:

- just fertiliser in the model represents a single common straight line for all 3 varieties,
- adding variety makes a set of 3 parallel straight lines and
- adding the interaction of fertiliser by variety allows the slopes to change, so making a set of 3 separate straight lines.

The mixing of numeric and categorical variables is achieved by using a linear model. The rationale for comparing models is to select the model giving the simplest, yet adequate summary of the observed data. Ideally the simplest model here is a single regression line, as it has only fertiliser in the pattern.

In Stata we first make a copy of the **`variety`** categorical variable as a new numeric variable named **varietyn** with:

```
. encode variety, generate(varietyn)
. codebook varietyn
```

The codebook command reveals that the numerical values created for **`varietyn`** have inherited the value labels of the original string variable, so **`1=NEW, 2=OLD, 3=TRAD`**. This extra step is necessary to obtain the breakdown of the ANOVA table into a hierarchical order of the competing models.

Now fit all terms into the model by:

```
. anova yield fertiliser varietyn fertiliser*varietyn, category(varietyn) sequential
```

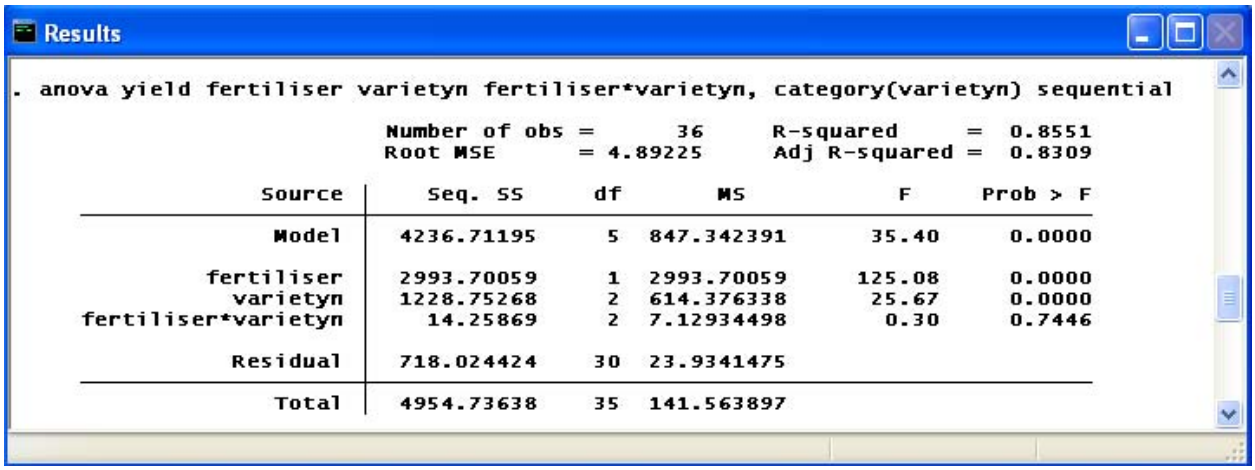
This gives the output shown in ***Fig. 16.3***.

Note:

(a) In our previous use of the anova command (see section 12.3 in chapter 12), we used the option ***continuous(<varname>)*** to indicate a continuous variable in the model. Here we do the opposite with ***category(<varname>)*** to indicate categorical variables. If neither is used, all terms in the model are assumed to be categorical.

(b) The option **sequential** fits terms one by one so that each term is adjusted for the preceding terms.

Fig. 16.3 Results of exploring effects of fertiliser and variety



In the output of ***Fig 16.3***, the rightmost column of the ANOVA table tests the effect of a term for its inclusion in the pattern. There is a strong effect of fertiliser and also a strong effect of varieties, but the two terms do not seem to interact. This leads us to choose the model with a set of 3 parallel straight lines. This model is the simplest, yet is an adequate enough summary, of the observed data.

16.4 Fitting the chosen model

Having used hypothesis tests to select between competing models, we now fit the chosen model, that is, we omit the interaction between fertiliser and variety from the pattern:

. anova yield fertiliser varietyn, category(varietyn) sequential

It may be observed that the residual MS (mean squares) terms in the ANOVA table do not change much because although the sums of squares explained by the interaction is now reabsorbed into the residual part, this is offset by the 2 extra residual degrees of freedom.

16.5 Checking the assumptions

Before presenting estimates and their measures of precision [standard errors] we must make sure that the assumptions upon which our linear model is based are sound. Else we risk interpreting parameters of a flawed model. Note that in general, no model is perfect. What we require is an adequate enough description of the data.

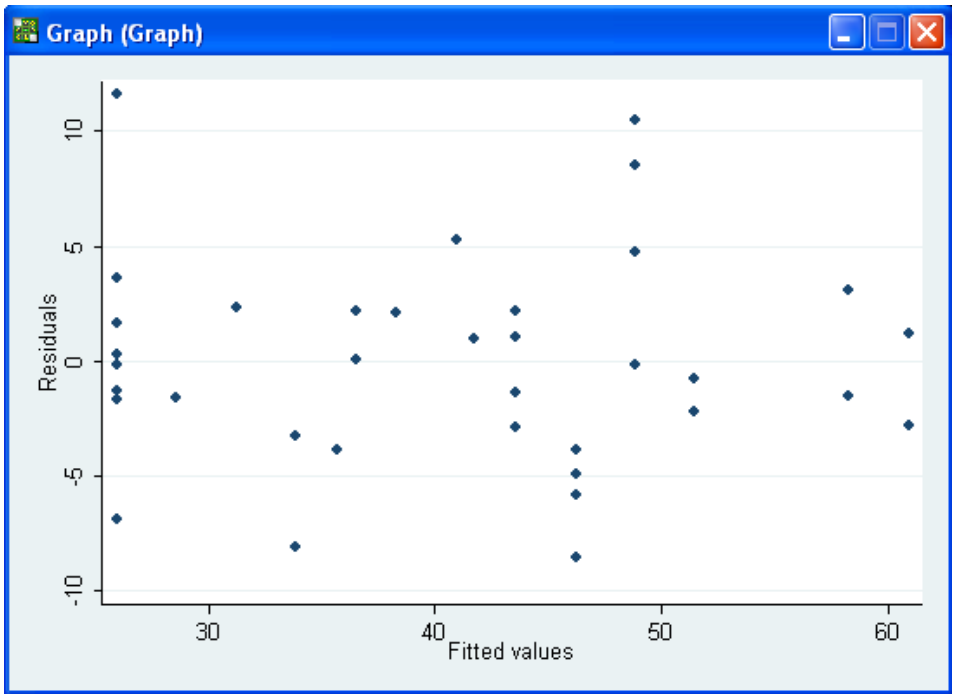
The modelling paradigm we adopted of **data = pattern + residual** requires the residual term to be normally distributed with constant variance. The really stringent assumption is that of constant variance.

Checks are done graphically as follows: as Stata stores the results of the last model we can use these immediately with:

. rvfplot

producing the scatterplot in ***Fig. 16.4***.

Fig 16.4 Plot of model residuals versus fitted values



For the residual term to have constant variance the plot of residuals against fitted/predicted values should show no obvious departures from a random scatter. ***Fig. 16.4*** shows no recognisable pattern, so the assumptions behind our model appear tenable and interpretation of its results is safe. We can now proceed to present estimates and their standard errors.

16.6 Reporting the results

The next step is to obtain the estimates of the 4 parameters of the regression lines, i.e. 3 separate intercepts and one common slope use:

. anova yield fertiliser varietyn, category(varietyn) regress noconstant

The resulting output appears in ***Fig. 16.5***: Ignore the ANOVA table with a single row for the combined model and focus on the table of parameter estimates with their 95% confidence intervals.

Note the use of the **regress** and **noconstant** options: while the former prints the table of parameter estimates of the linear model, the latter gives these parameters as absolute values instead of differences from a reference level, as is done by default in most statistics software. Absolute values are useful here in order to present 3 predictive equations, one for each variety.

From ***Fig 16.5***, the equations for predicting yield of each variety are:

`yield of NEW variety = 47.75 + 5.26x`

`yield of OLD variety = 35.69 + 5.26x`

`yield of TRAD variety = 25.96 + 5.26x`

where x is a set amount of fertiliser in the observed range of 0 to 3 units.

The intercept is the estimated yield of each variety at x=0, i.e. when no fertiliser is applied. The increase in yield for each 1 extra unit of fertiliser applied is estimated at 5.26 yield units. Finally, we are 95% confident that the range 3.3 to 7.2 yield units contains the true value of the rate of increase, which is common to all 3 varieties.

Fig 16.5 Results from use of anova command

Results

```
. anova yield fertiliser varietyn, category(varietyn) regress noconstant

      Source |       SS       df       MS              Number of obs =      36
-------------+------------------------------           F(  4,    32) =  693.79
       Model |  63506.5867     4  15876.6467            Prob > F      =  0.0000
    Residual |  732.283114    32  22.8838473            R-squared     =  0.9886
-------------+------------------------------           Adj R-squared =  0.9872
       Total |  64238.8698    36  1784.41305            Root MSE      =  4.7837

------------------------------------------------------------------------------
       yield |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
fertiliser |   5.264286   .9554869     5.51   0.000     3.318023    7.210549
  varietyn |
           1 |   47.75536   3.216022    14.85   0.000     41.20453    54.30618
           2 |   35.68676   2.116752    16.86   0.000     31.37508    39.99845
           3 |   25.96405   1.436038    18.08   0.000     23.03893    28.88916
------------------------------------------------------------------------------
```

Now make a new variable **fitted** which stores the predicted values of yield according to the parallel lines model above. Since Stata stores information from the last model fitted, it is simply:

. predict fitted

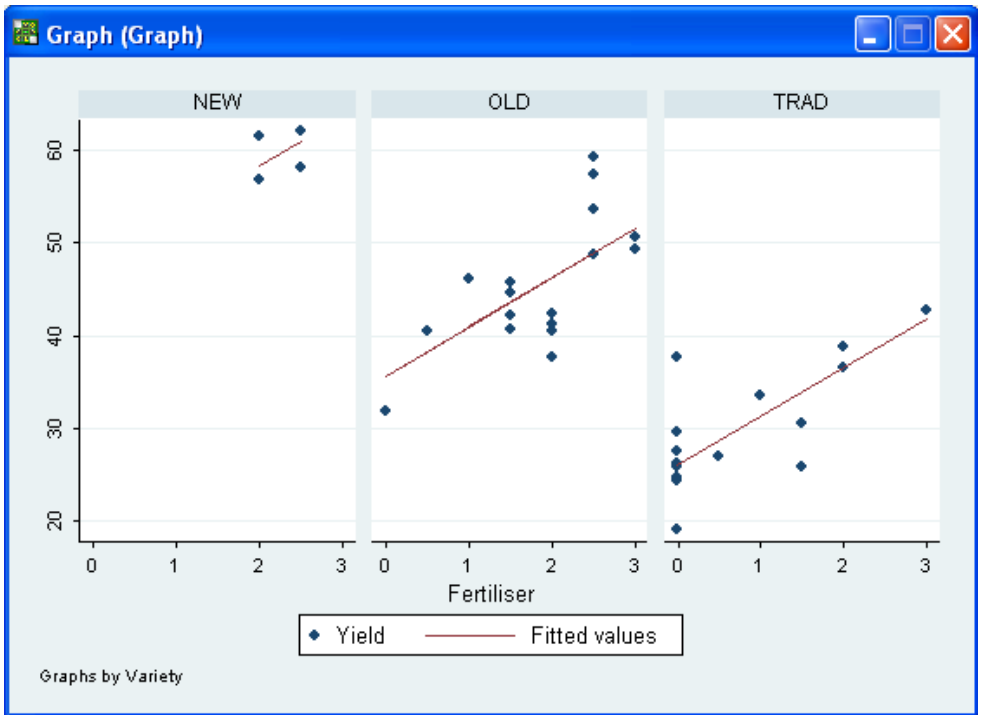
Finally, we create ***Fig. 16.6*** which illustrates the fitted model with:

. twoway (scatter yield fertiliser) (line fitted fertiliser), by(variety, rows(1))

The same graph can be produced using, instead of brackets, the graph separator || as shown below:

. scatter yield fertiliser || line fitted fertiliser , by(variety, rows(1))

Fig. 16.6 Parallel regression lines fitted to each variety



16.7 Adding further terms to the model

We have illustrated the principle of statistical modelling by building a linear model with just two of the four potential factors which we thought may affect rice yields. The two factors we disregarded were village and size (in acres) of the field. It would be possible to assess the importance of village in the same manner as we explored variety, by just adding village into the pattern with (say):

. encode village, generate(villagen)

. anova yield villagen fertiliser varietyn, category(varietyn villagen) sequential

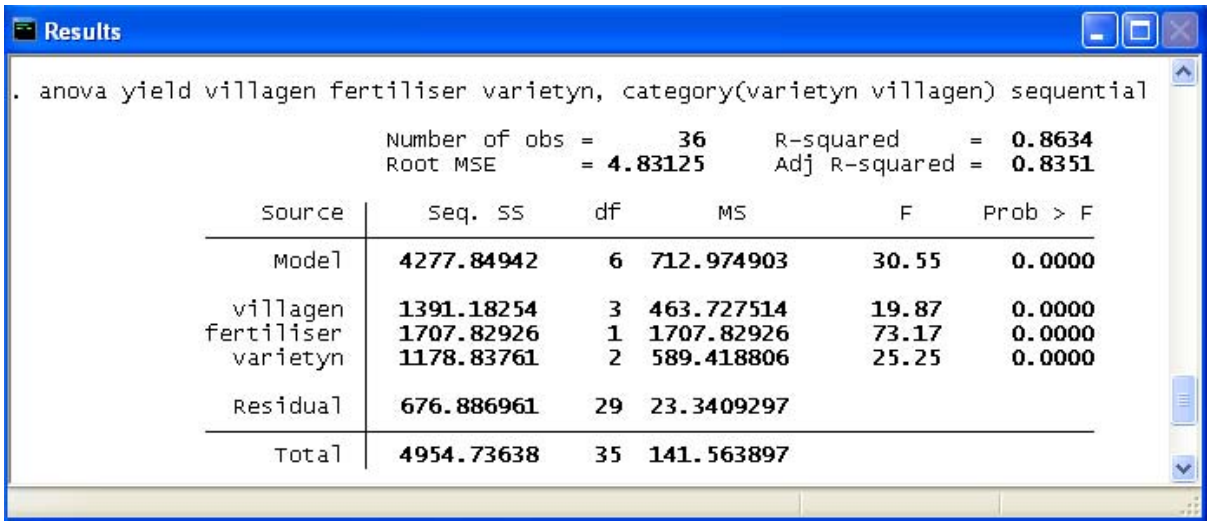
and then assessing the effect of village in the same way as it was done in ***Fig. 16.2***.

The principle of assessing the effect of village before other factors is that of accounting for the variability observed in yield of those factors that are uncontrollable. In this context, village is just the geographical location, so its effect must be discounted before assessing the effect of other factors like fertiliser and variety over which there is some control.

The output from the above command (see ***Fig. 16.7***) shows that fertiliser and variety still have an effect on yield after allowing for variability between villages.

Likewise, the size of the field can be investigated as a continuous variable. Recall the previous command and try incorporating it as the last term in the model. What do you conclude? Is size an important contributor to variation in rice yields?

Fig 16.7 Effect of adding village to the model of yield by fertiliser and variety



16.8 Use of regress as an alternative to anova

It is possible to reproduce an equivalent analysis to the one above with the **regress** command, using the **xi** prefix to create indicator variables for categorical columns, e.g.

. xi: regress yield fertiliser i.variety i.variety*fertiliser

Note that the prefix **xi:** requires all categorical variables in the regress command to have the prefix **i.** .

The output of the regress command is different from that of the anova command in that the ANOVA table is not broken down into rows testing the effect of adding one extra term in the pattern over and above those already present, as illustrated in ***Fig. 16.3*** and ***Fig. 16.7***. Nor is it

possible to obtain the absolute value of the parameter estimates as shown in ***Fig. 16.5***. To illustrate, we present in ***Fig. 16.8***, the results of the model fitted in ***Fig. 16.7*** using:

. xi: regress yield i.village fertiliser i.variety

These results show the correct overall Model SS, but not the SS separately for village, fertiliser and variety. To obtain these, it is necessary to use the test command as follows:

. test _Ivariety_2 _Ivariety_3

The results are shown in ***Fig. 16.9***. They coincide with results for variety shown in ***Fig. 16.7*** above.

Why introduce the regress command here? Because in Stata, the ***anova*** command does not allow for sampling weights, i.e. the `pweight` option is not allowed, only `aweight` and `fweight`. Hence if the regression analysis is to be done properly using the appropriate sampling weights, then the ***regress*** command above has to be used. This is discussed in the following section.

Fig 16.8 Results using the regression command with prefix xi:

Results

```
. xi: regress yield i.village fertiliser i.variety
i.village         _Ivillage_1-4       (_Ivillage_1 for village==KESEN omitted)
i.variety         _Ivariety_1-3       (_Ivariety_1 for variety==NEW omitted)

      Source |       SS       df       MS              Number of obs =      36
-------------+------------------------------           F(  6,    29) =   30.55
       Model |  4277.84942     6  712.974903            Prob > F      =  0.0000
    Residual |  676.886961    29  23.3409297            R-squared     =  0.8634
-------------+------------------------------           Adj R-squared =  0.8351
       Total |  4954.73638    35  141.563897            Root MSE      =  4.8312

------------------------------------------------------------------------------
       yield |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
  _Ivillage_2 |    3.700191   2.471155     1.50   0.145    -1.353889    8.754271
  _Ivillage_3 |    1.892548   3.000153     0.63   0.533    -4.243454    8.028549
  _Ivillage_4 |    3.316521   2.594134     1.28   0.211     -1.98908    8.622121
   fertiliser |    4.590734   1.242909     3.69   0.001       2.0487    7.132768
  _Ivariety_2 |   -11.54938   2.744873    -4.21   0.000    -17.16328   -5.935487
  _Ivariety_3 |   -21.55727   3.112564    -6.93   0.000    -27.92317   -15.19136
        _cons |    45.76249   3.761108    12.17   0.000     38.07016    53.45482
------------------------------------------------------------------------------
```

Fig 16.9 Results of an overall test for comparing variety levels

Results

```
. test _Ivariety_2 _Ivariety_3

 ( 1)  _Ivariety_2 = 0
 ( 2)  _Ivariety_3 = 0

       F(  2,    29) =   25.25
            Prob > F =    0.0000
```

16.9 Using sampling weights in regression

We illustrate the use of sampling weights in regression using the same paddy survey data, but now taking account of the sampling structure. The 36 observations in the data file **paddyrice.dta** were the results of a crop-cutting survey undertaken in a small district having 10 villages. The 10 villages had (respectively) 20, 10, 8, 30, 11, 24, 18, 21, 6 and 12 farmers, each with one field on which to grow rice. Thus there were a total of 160 farmers (fields) in the district.

Let us first suppose that the 36 fields for which information is available in **paddyrice.dta** were selected at random from the 160 fields available. The sampling weight for each of the 36 fields is then 160/36 = 4.444, this being the inverse of the probability of selecting a field. Open the data file named **paddy_weights.dta**. This data file has weights already included.

The following command allows this sampling weight to be incorporated in the regression model fitted in ***Fig. 16.8***.

. xi: regress yield i.village fertiliser i.variety [pweight=srs_wt]

The results are shown in ***Fig. 16.10*** and demonstrate that the model parameters (Coef.) do not change. This is because the unweighted analysis also assumes simple random sampling, but from an infinite population. However, the standard errors are different, and these have taken account of the sampling weights.

Fig 16.10 Exploring effects of village, fertiliser and variety, with srs sampling weights

Results

```
. xi: regress yield i.village fertiliser i.variety [pweight=srs_wt]
i.village         _Ivillage_1-4       (_Ivillage_1 for village==KESEN omitted)
i.variety         _Ivariety_1-3       (_Ivariety_1 for variety==NEW omitted)
(sum of wgt is   1.6000e+02)

Linear regression                               Number of obs =      36
                                                F(  6,    29) =   66.52
                                                Prob > F      =  0.0000
                                                R-squared     =  0.8634
                                                Root MSE      =  4.8312
```

yield	Coef.	Robust Std. Err.	t	P>|t|	[95% Conf.	Interval]
_Ivillage_2	3.700191	2.476394	1.49	0.146	-1.364604	8.764986
_Ivillage_3	1.892548	2.06081	0.92	0.366	-2.322281	6.107376
_Ivillage_4	3.316521	2.313136	1.43	0.162	-1.414374	8.047415
fertiliser	4.590734	1.221201	3.76	0.001	2.093098	7.08837
_Ivariety_2	-11.54938	2.053313	-5.62	0.000	-15.74888	-7.349887
_Ivariety_3	-21.55727	2.04535	-10.54	0.000	-25.74048	-17.37406
_cons	45.76249	2.741228	16.69	0.000	40.15605	51.36893

Now let's suppose that the sample was selected in the following way.

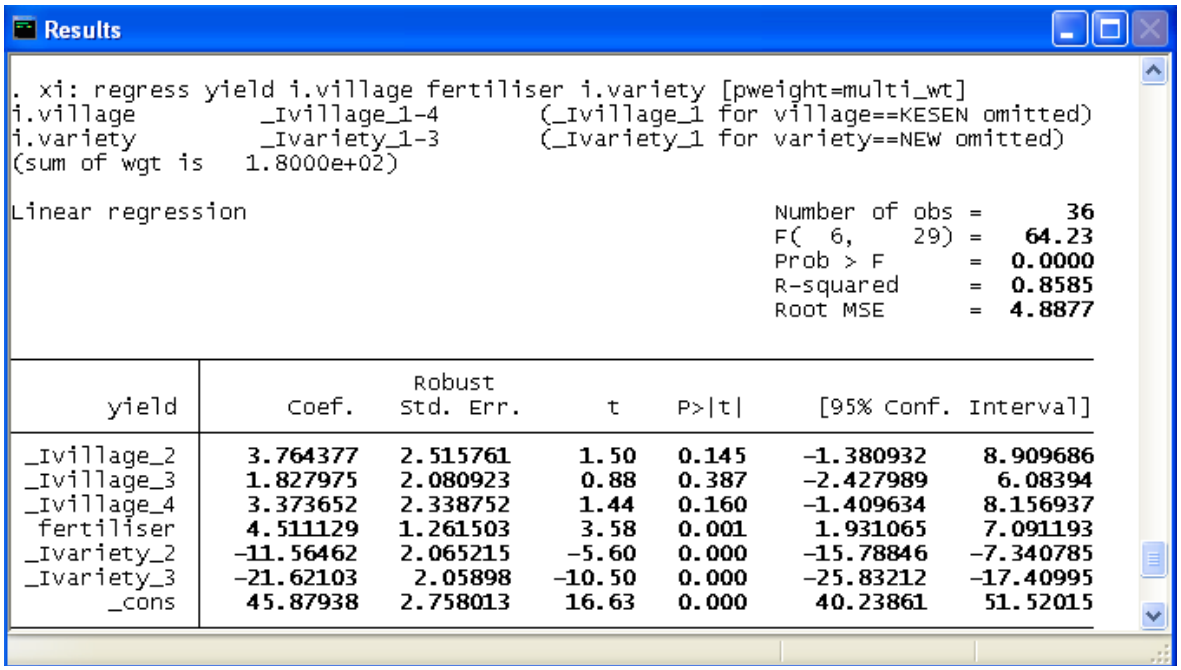
- First, 4 villages were selected from the 10 villages with simple random sampling. The villages selected were village numbers 1, 2, 4 and 10 having 20, 10, 30, 12 fields respectively.
- At the next stage of sampling, 10, 5, 14 and 7 fields were selected from each of these villages with simple random sampling.

The sampling weights resulting from the scheme above are found in the variable called ***multi_wt*** in the data file **paddy_weights.dta**. Recall the previous regress command, and change the weight variable to be ***multi_wt*** as in the following command.

. xi: regress yield i.village fertiliser i.variety [pweight=multi_wt]

The results are shown in ***Fig. 16.11***. Notice now that both the estimates as well as the standard errors differ. The differences in this particular example are however very minor.

Fig 16.11 Exploring effects of village, fertiliser and variety, with multistage sampling weights



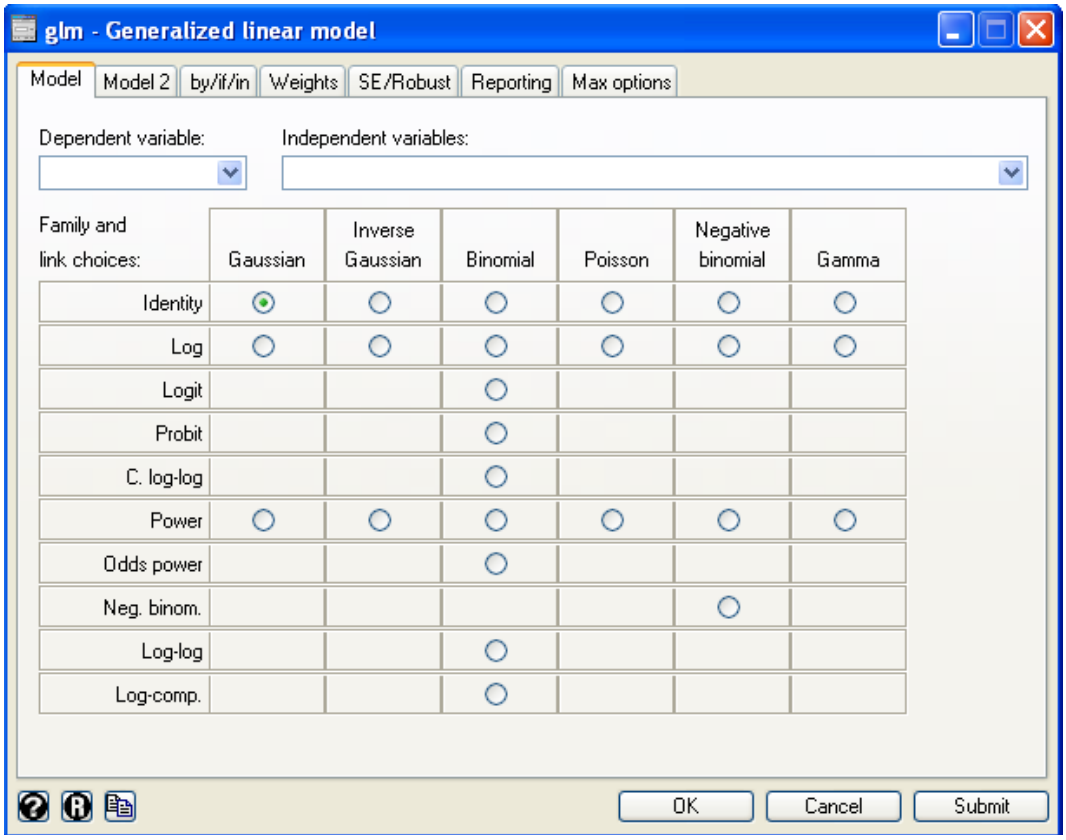
16.10 Extending the modelling approach to non-normal data

A common example of non-normal data arises with proportions, e.g. in this dataset, the proportion of farmers who have planted the **NEW** variety of rice.

Stata can be used to extend the modelling approach to data that are non-normal by using the Generalised Linear Models (GLM) framework.

Currently, Stata features 5 non-normal distributions, as shown in ***Fig. 16.12***, obtained using the menu sequence **Statistics ⇒ Generalized Linear Models (GLMs) ⇒ Generalized Linear Models (GLMs)**. Note that Gaussian is a synonym of "normal".

Fig 16.12 Dialogue for modelling with non-normal distributions



Chapter 17 Tailoring Stata

The next three chapters consider possible strategies for using Stata.

With spreadsheets often everyone uses them, but no one is a real expert. In contrast, if an organisation uses data management software, then there will usually be a person or team with more expertise who constructs the databases. Then the rest of the staff make use of them, and perhaps write reports.

With Stata in an organisation it would be sensible if there were a similar split, with a small group developing the expertise to support effective use of the software. Other staff need to understand enough so they know what to ask for. In this guide we provide this minimum. There is a guide on programming in Stata for those who wish to learn more.

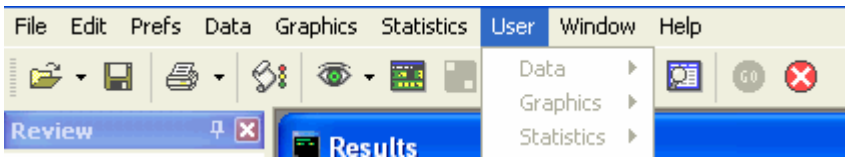
If you are using Stata alone, you will find there is an active group of Stata enthusiasts round the world, who could help if you require advice about facilities that are not in the current version. This guide is designed to give you the understanding so you could communicate with this group, and take advantage of their suggestions.

This chapter outlines how users can add to the Stata's menu system and also how they can add their own help files. These are both easy steps, that illustrate the philosophy of Stata. It is an open system that encourages users to tailor the software so it becomes convenient for their applications.

17.1 Adding to the menus

Stata has a menu called **User**, see ***Fig. 17.1***.

Fig. 17.1 Stata's User menu



It includes three items, called **Data**, **Graphics** and **Statistics**, that parallel the main menus in Stata, see ***Fig. 17.1***. However, nothing happens when you click on the items in the **User** menu. It is easy to change this, and add your own menus.

You can choose either to extend the User menu itself, or to add submenus. For illustration, consider the facilities in Stata to find duplicate observations in a dataset. They are available under the **Data ⇒ Variable utilities** menu, which gives the options shown in ***Fig. 17.2.***

Fig. 17.2 Data ⇒ Variable Utilities

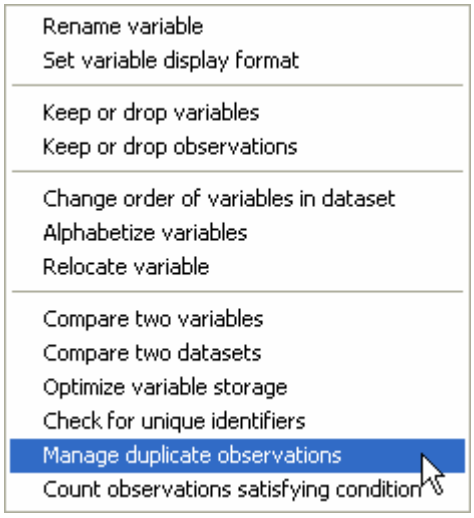
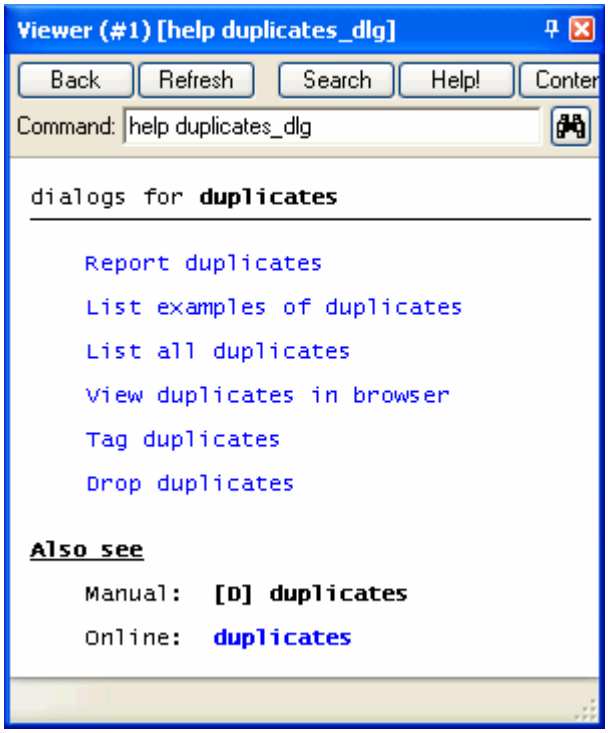


Fig. 17.3 Facilities for duplicates



Clicking on the item to **Manage duplicate observations** does not give a dialogue directly. Instead it loads the viewer, shown in ***Fig. 17.3***. You can click on any of these items to provide the appropriate dialogue.

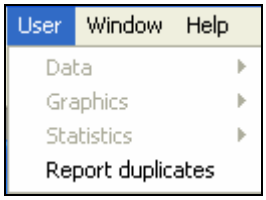
If you often use these facilities, then perhaps they could be made more accessible, via the User menu.

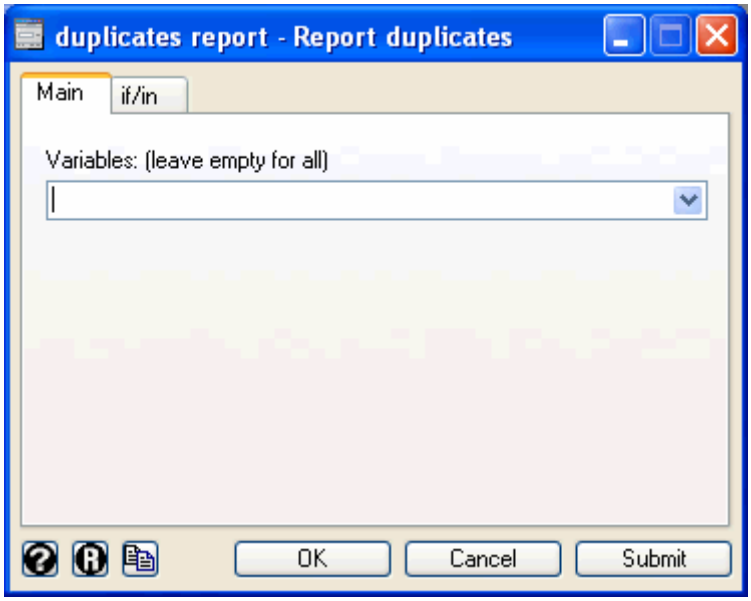
Adding to the user menu is easy. For example, type the command:

. window menu append item "stUser" "Report duplicates" "db duplicates report"

Now the user menu is as shown in ***Fig. 17.4,*** and clicking on the new item, produces the dialogue directly, also shown in ***Fig. 17.4***.

Fig. 17.4 User menu with one item added to produce the dialogue directly

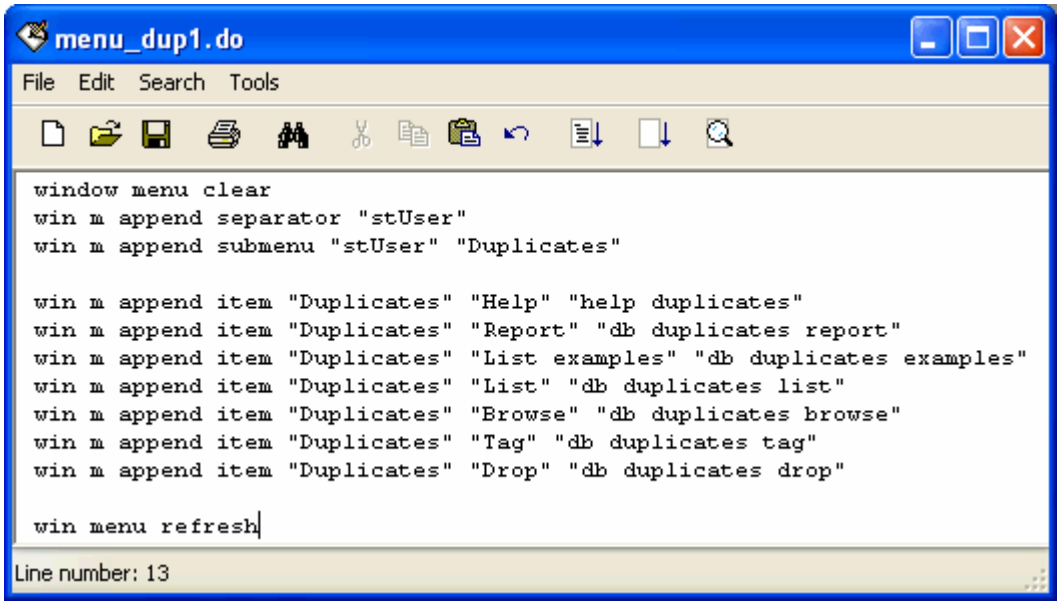




The structure of the command, is as follows. An item is appended to the existing menu, which is called **stUser**. (The other menus are called **stUserData**, **stUserGraphics**, and **stUserStatistics**.) The text to appear is the phrase “Report Duplicates”, as shown in ***Fig. 17.4***. When this item is clicked it generates the action, “**db duplicates report**”, which is the instruction to load the duplicates report dialogue, see ***Fig. 17.4***.

A do file is used to construct the full menu for duplicates. One possibility is shown in ***Fig. 17.5***

Fig. 17.5 Do file to add to the User menu



The commands are simpler to follow, when you see what it has done. The menu, after running this file is shown in ***Fig. 17.6***.

The first command in ***Fig. 17.5*** clears any existing menu items. Then a separator is added, followed by a submenu, that we call **Duplicates**, see ***Fig. 17.6***. Items are now appended to this submenu. The first gives the Stata help on duplicates, as shown in ***Fig. 17.7***. The remaining items give the alternative dialogues to examine duplicate observations.

Fig. 17.6 New User menu

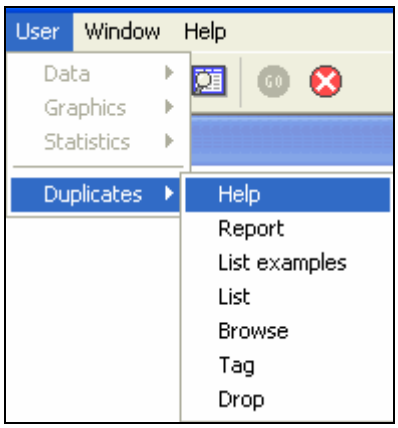


Fig. 17.7 Help on duplicates from the user menu

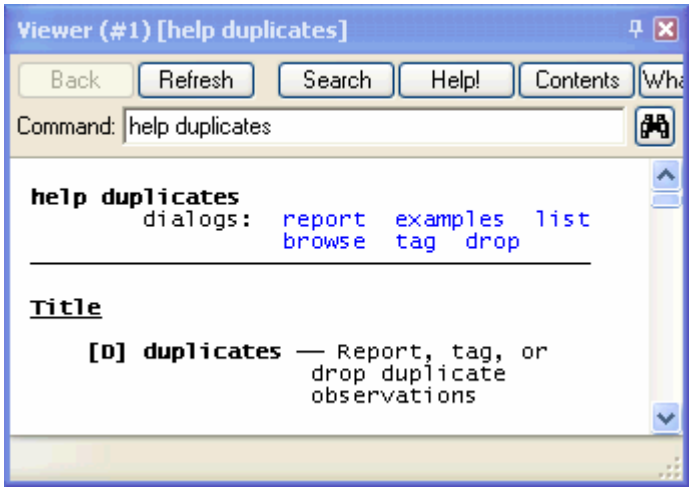


Fig. 17.8 and ***17.9*** show other layouts for the menu items. Only two changes are needed to the **do** file in ***Fig. 17.5***, to produce the layout in ***Fig. 17.8***. The first is to delete or comment-out the second line, that gives the separator. The second is to replace **stUser** in line 3, with **stUserData**.

The menu layout in ***Fig. 17.8*** makes it clear that **duplicates** is a data-type utility, but there are more clicks to make. The opposite extreme is shown in ***Fig. 17.9***, which puts all the items on the main User menu.

Fig. 17.8 Alternative layout for User menu

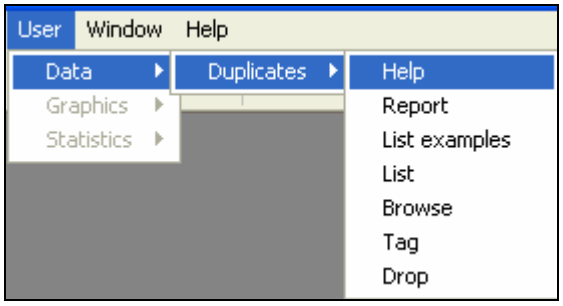


Fig. 17.9 Another layout

17.2 Adding help files

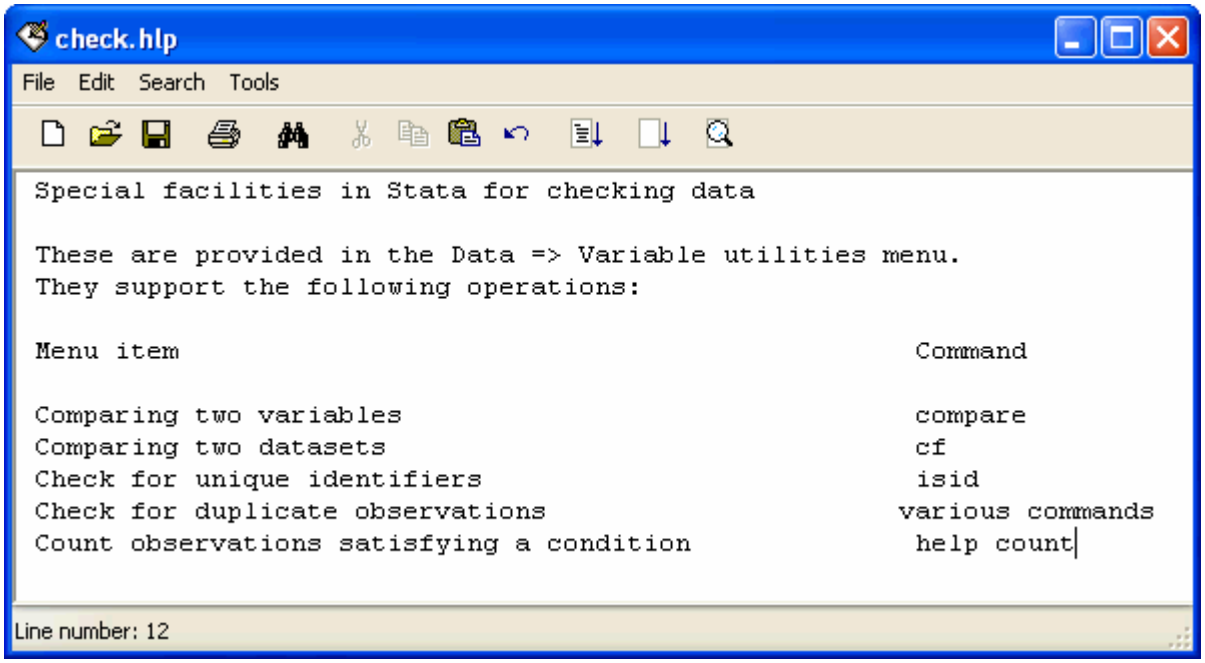
Help files are usually to give information on Stata's commands, but they can be used more generally as is shown in this section. We propose to give some information on Stata's facilities for checking data. The file must be given a name, and it must have the extension hlp. We propose the name **`check.hlp`**.

First verify that the name check is not already a Stata command. This can be done by typing the command

. check

Stata responds by saying this is an unrecognised command, which is what we want. Type some text into any editor, like notepad, or use Stata's do-file editor. The text we started is in ***Fig. 17.10***.

Fig. 17.10 A new help file



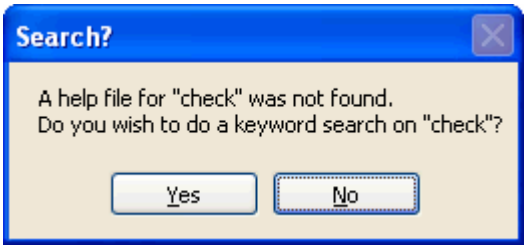
Having saved the file, type:

. help check

to see the contents in Stata's viewer.

If this does not work, it may be that Stata does not recognise the current working directory. In that case you would get a message such as is shown in ***Fig. 17.11.***

Fig. 17.11 Stata response if it cannot find the help file



In that case type the Stata cd command (for change directory). When we did:

. cd

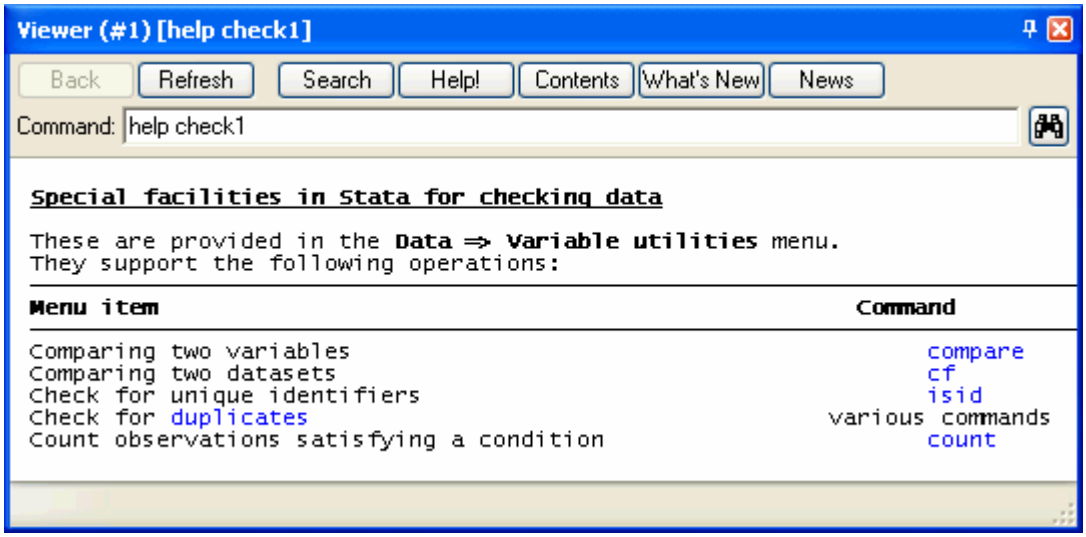
Stata responded with **C:\DATA**, which was certainly not our current directory. In Chapter 19 we will show how to change the working directory permanently, but for now just use the **cd** command again with your current directory. For us it was as follows:

. cd "C: \Administrator\My documents\Stata guide\Files\"

Then try the help command again.

As usual with Stata, you may want to go a little further, and make the help file more impressive. ***Fig. 17.12*** shows the text from ***Fig. 17.10***, but displayed in roughly the same way as other Stata help.

Fig. 17.12 Making the help file consistent with other Stata help



To do this, use the command **{smcl}**, on the first line of the help file. The letters smcl stands for Stata Markup and Control Language. This allows you to put commands with the text in curly brackets, as in ***Fig. 17.13***. Some of the following were used:

{.-} to give a line of dashes

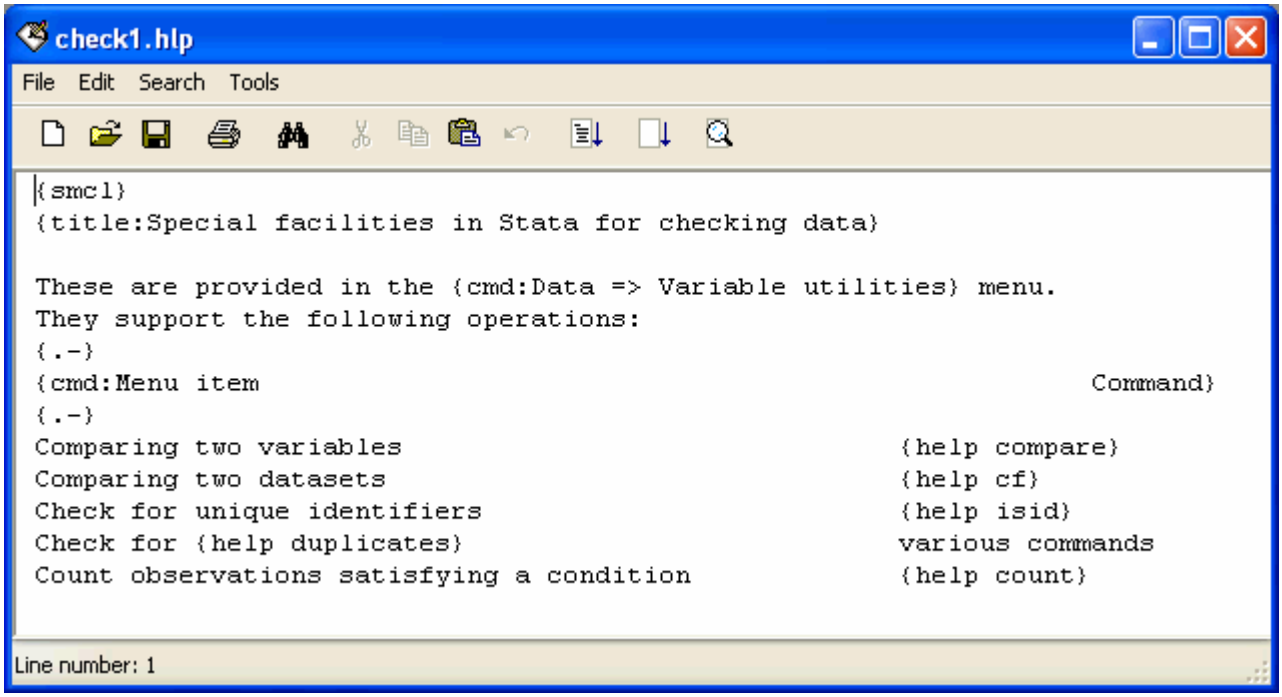
{Title: …} to format the remaining text in the curly brackets however Stata chooses to format titles. In ***Fig. 17.12*** they are in bold and underlined.

{cmd: any command} to format the text as Stata does for commands, namely in bold.

{it: any text} to format the text as italic

{help any Stata command} to give a hypertext link to the help on that command. In ***Fig. 17.12***, if you click on the word **count**, (underlined in blue on colour screens) Stata provides the help on the **count** command.

Fig. 17.13 Help file using Stata's smcl commands



The only remaining task is to inform users of the name of the help file. One way is to use the tools from Section 17.1 and type something like:

. window menu append item "stUser" "Information on checking" "help check"

this adds the help file to the User menu.

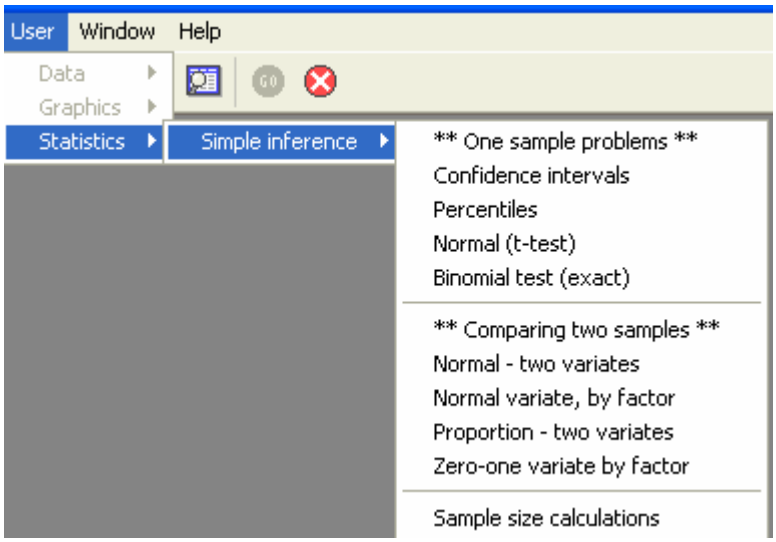
17.3 Stata on training courses

Statistics packages are often used in support of training courses, and the facilities in Stata described in this chapter can enhance the way training can be provided.

A recent training course reviewed the basic concepts of inferential statistics. This topic is used as an example of the type of menu that could be added to Stata.

Two help files were first prepared, one to describe the use of Stata for **One-sample problems**, and the other for **Two-sample problems**.

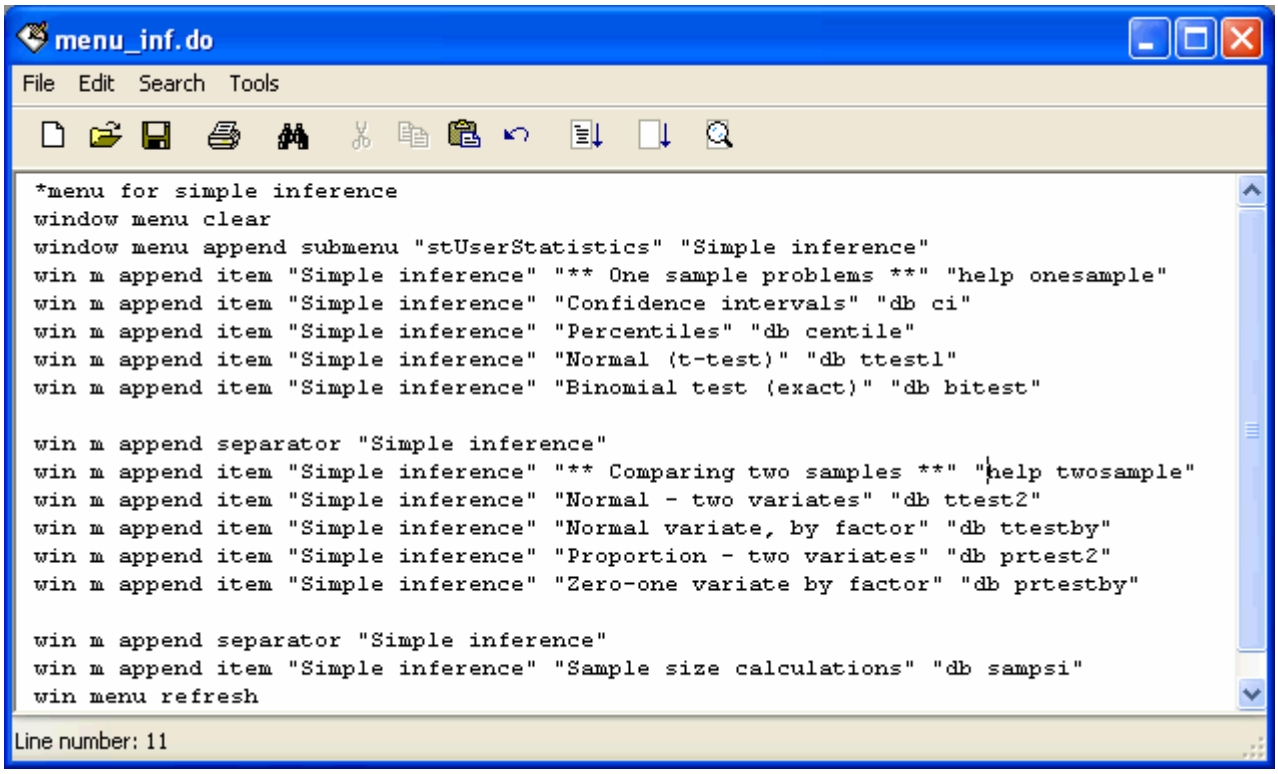
Fig. 17.14 A user menu to support teaching



Then a special menu was prepared, that collects the dialogues together, and is in line with the way the course was taught. The dialogues all exist already, under the statistics menu, but they are scattered. Other similar dialogues in the full menu system could also distract participants from the topics covered in the course.

The do file to produce the menu in ***Fig. 17.14*** is given in ***Fig. 17.15***.

Fig. 17.15 Do file for the teaching menu



17.4 In conclusion

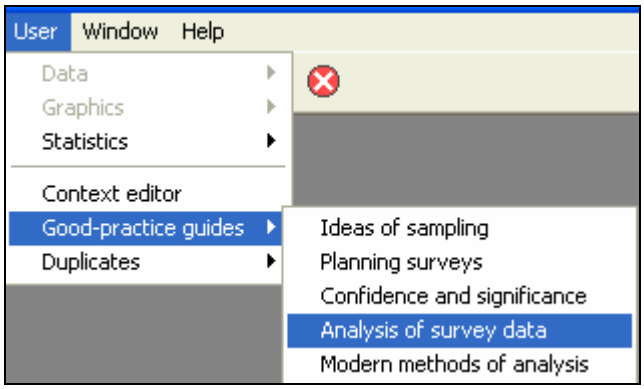
Some readers will have been surprised at how easily they can add to Stata's menus and help files. They would have assumed that such changes require a "real programmer".

Our aim remains one of alerting Stata users as to what is possible, rather than changing them into programmers. More about how Stata can be tailored in the next chapter.

To finish this chapter two further applications of the topics covered here are described. The first is to add documentation for reference purposes or to support training courses. An example uses some good-practice guides that were prepared at Reading, for researchers involved in conducting surveys or experiments. Nineteen such guides were prepared, covering design, data management, analysis and presentation of the results. They can be downloaded from the web site, www.ssc.rdg.ac.uk.

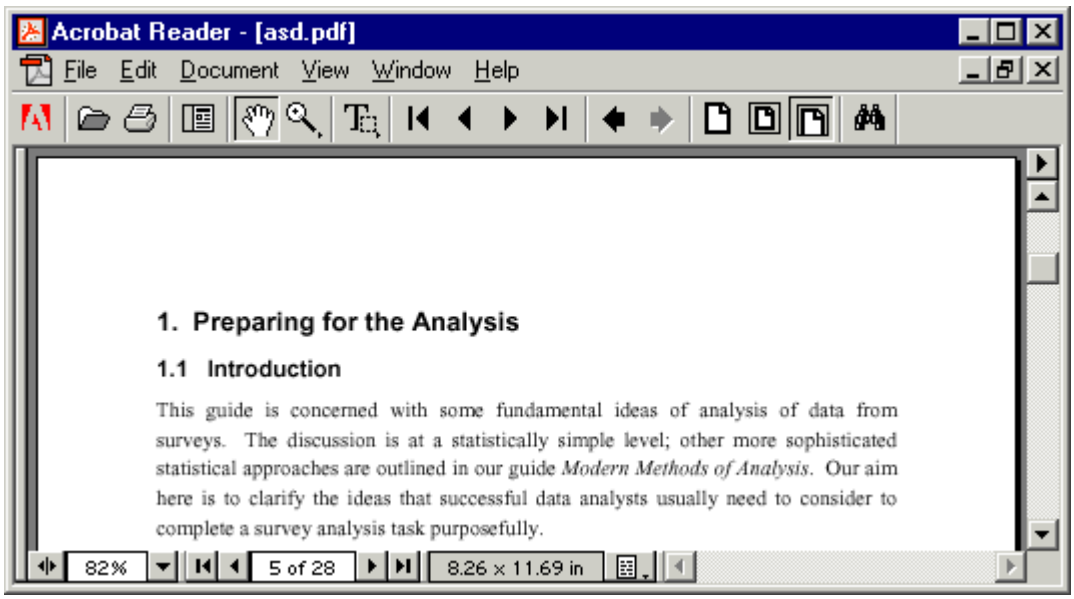
They are available as "pdf" files, and can therefore be read using the Acrobat reader, available from adobe, on www.adobe.com.

Fig. 17.16 Adding good-practice guides to Stata's menus



The call to some of these guides has been added to Stata's user menu, using the same ideas as were explained in Section 17.2. The appropriate commands were added to the file prepared earlier, see ***Fig. 17.5***.

Fig. 17.17 Part of one of the good-practice guides



The second development was to look for an improved editor to use with Stata. We are quite happy with the do-file editor provided within Stata, but sometimes found that a more powerful editor would be useful. The Stata user community has an article titled "Some notes on text

editors for Stata users". This is available at http://ideas.repec.org/c/boc/bocode/. From this list we found and downloaded a free editor, called ConTEXT. This editor can also be called from our new Stata menu, see ***Fig. 17.16***.

There are three ways in which a more powerful editor may be of use. The first is to write do files, or the ado files that are introduced in Chapter 18. The second is to edit ASCII data files. These may be large, and modern editors can handle files of 10's of megabytes. ConTEXT can, for example, mark and copy columns within a file, which is sometimes useful. Thirdly we could edit the results, for example tables, prior to passing them to a Word processor. There are options to export files from the editor in either HTML, or in RTF formats.

The commands to access these items and to add them to the menu are provided in the file called **menu_gsp.do**. As written they do depend on the file locations of the supporting files. One of the lines from **menu_gsp.do** is as follows:

win m append item "Good-practice guides" "Analysis of survey data" `"winexec "C:\Program Files\Adobe\Acrobat 6.0\Reader\AcroRd32.exe" "C:\ado\asd.pdf""'

Where Stata is used by an organisation, it is of benefit if some users, or a group, develop expertise to streamline the use of the software for everyone. These two additions provide examples of the value of an organised approach. Often an institute, or a training course will have a small set of documents that could useful be added to the menu, as shown in ***Fig. 17.16***. It would be simple if they were in the same directories on each machine, or centrally on the network. Then the same do file can be used to add them to each user. Similarly, if an organisation decides to use a more powerful editor, then work would be simpler, if they agree on one that particularly suits their needs.

Chapter 18 Much to ADO about

In Chapter 5 we explained why it is important for survey analysis to keep **do** files as a record of the analyses, rather than just working from the menus. In this chapter the **do** file is generalised into an **ado** file.

One of the strengths of Stata is the ease with which **do** files can be constructed and then generalised into **ado** files. An **ado** file is a set of Stata commands that can be passed to someone else. It adds a new command to the Stata language.

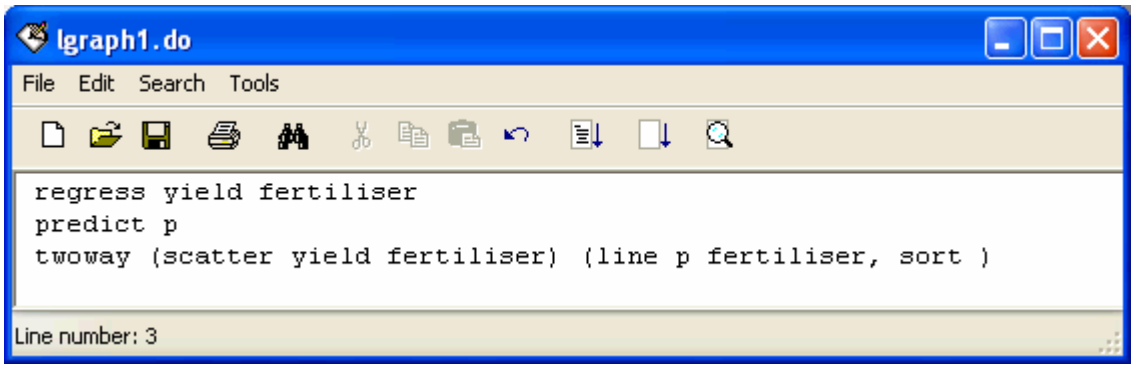
We are not trying to turn you into programmers. But we are trying to make it easier for you to communicate with programmers, or with the Stata enthusiasts, who have developed programs (**ado** files) themselves. When you discuss whether a feature can easily be changed, then it is useful if you have some idea whether you are suggesting work that will take perhaps an hour, or might be three months.

Also, as with the last chapter, some users will be surprised how easy it is to make modest changes themselves.

18.1 Starting with a do file

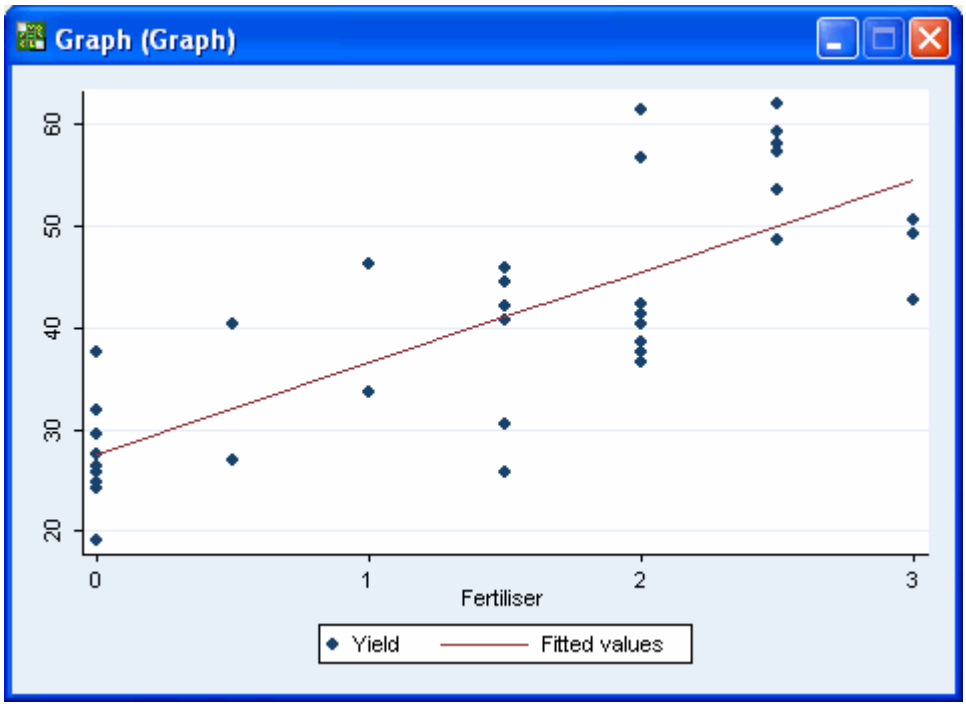
We follow the same process as Hills and Stavola (2004), by starting with a simple **do** file that adds a straight line to a scatter plot. Open the **`survey.dta`** worksheet and construct the **do** file shown in ***Fig. 18.1.***

Fig. 18.1 A simple do file



Running this file gives the graph shown in ***Fig. 18.2***

Fig. 18.2 Results from the do file in Fig. 18.1



If instead, you get an error, check the code and make a correction. Then before running the **do** file again also type

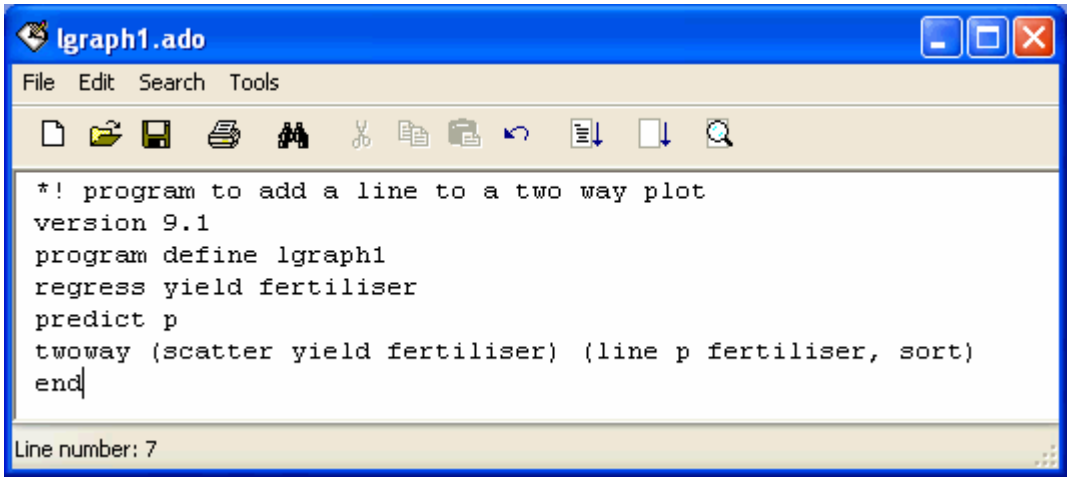
. drop p

as a command. Otherwise the program cannot run, because **p** cannot be created twice.

18.2 Making the do file into an ado file

Take the code in ***Fig. 18.1*** and add the extra commands shown in ***Fig. 18.3***.

Fig. 18.3 A simple ado file



Here the first line is a comment. The second is optional, and states which version of Stata was used to create this program. The third line states that a new Stata command, called **lgraph1** is defined.

The last line is the end of the program.

Save the file, and call it **lgraph1.ado**. The name of the file must be the same as the name used in the second line of ***Fig. 18.3***, which defines the program. Notice that the extension given to the file name is **ado** and not just **do**.

Type the command

. drop p

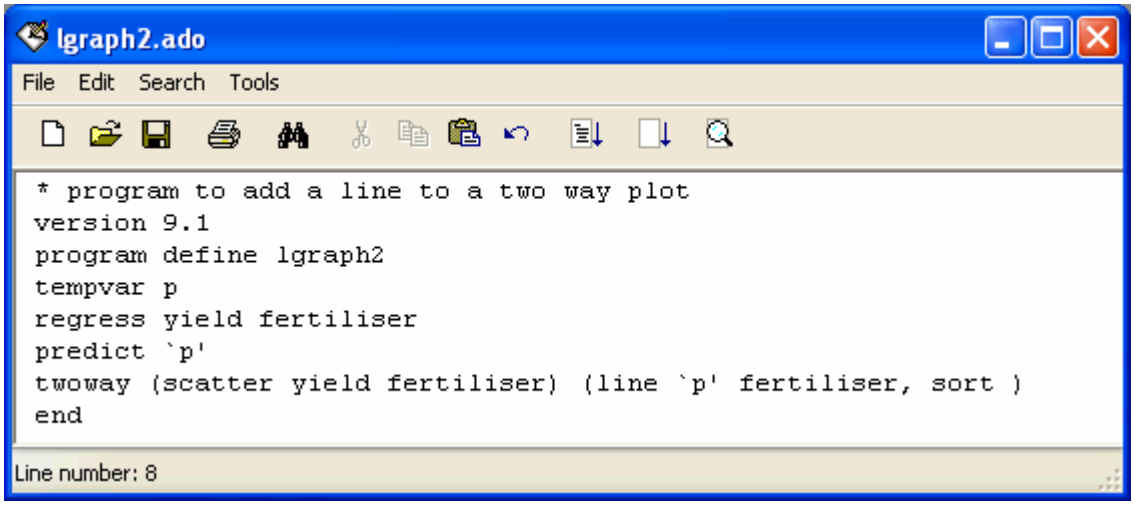
Now type the new command

. lgraph1

It should run and give you the same result as before.

Now the new command is improved, in stages. The first annoyance is that you continually have to type the command **drop p** between runs of the program. This is rectified in ***Fig. 18.4***.

Fig. 18.4 First improvement to the ado file



The extra line is the 4th one, where we say that **p** is a temporary variable. We also give this command a new name, so change the second line to **lgraph2**, and save the file as **lgraph2.ado.**

Now try the command by typing

. lgraph2

If it works, then try again, typing

. lgraph2

You no longer have to worry about dropping the variable between runs.

In summary an **ado** file, ***Fig. 18.3*** and ***Fig. 18.4*** does not look that different from a **do** file, ***Fig. 18.1***. The difference is however caused by the two lines

. program define <name of command>

…

. end

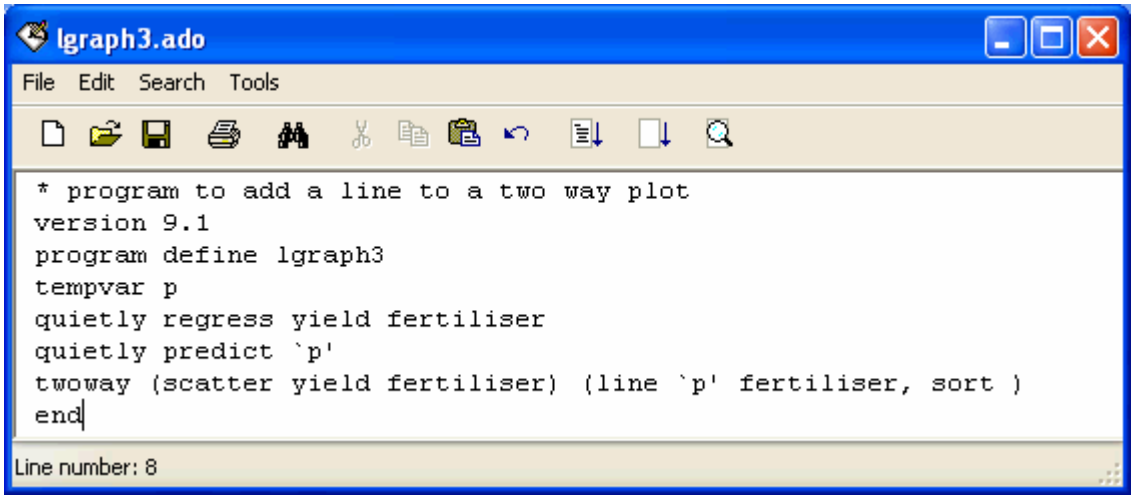
With these two lines the program defines a new command that you can use, rather than just running the set of commands. Both **do** files and **ado** files are useful, but they are different.

18.3 Cutting out unwanted output

The output that comes with the regress command may not be wanted, because regress is mainly used to get the predicted values. The prefix **quietly** before a command prevents all

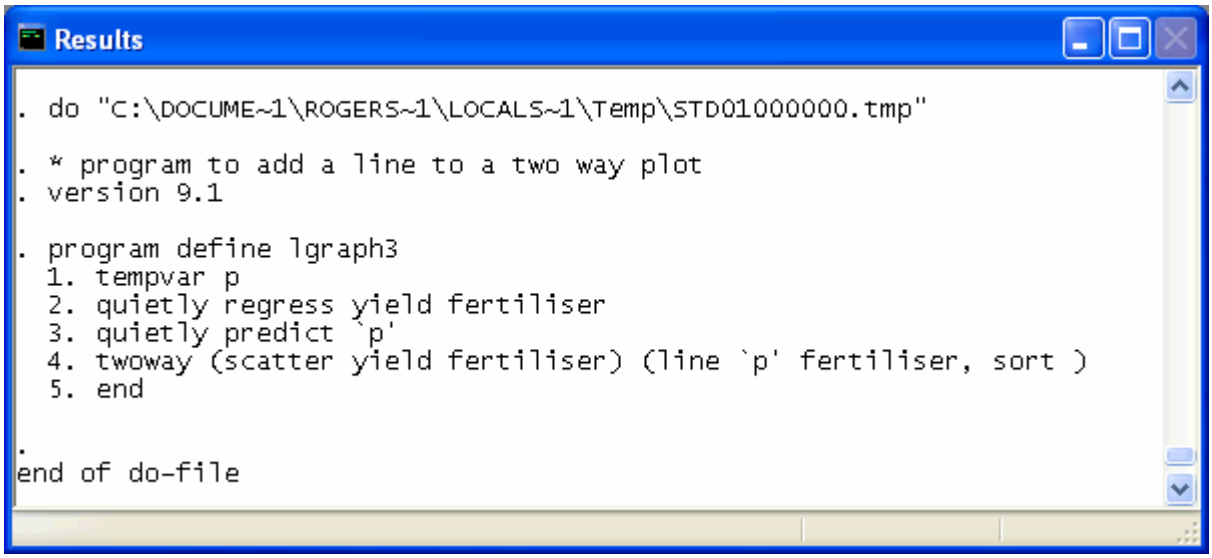
output except error messages. We have made this change in ***Fig 18.5***, and also changed the name of the file to **lgraph3**.

Fig. 18.5 Peventing output



This time try running the program file before saving it. This does not produce a graph, but the results window is roughly as in ***Fig. 18.6***.

Fig. 18.6 Results window after summary on ado file



What Stata has done is to define the new command, called **lgraph3**. This is now available for this session of Stata. So you can now type

. lgraph3

to get the graph, hopefully without the regression output in the results window.

If you need to correct, or improve the command, then you can make corrections in the editor in the usual way. But try running the code again and Stata gives an error. It says

lgraph3 already defined

What you must do is drop the program from Stata's memory, using

. program drop lgraph3

Once you are happy that the new program works, save this file as **lgraph3.ado**. Then try the new command again by typing:

. lgraph3

18.4 Making the program accept arguments

The program is currently only able to plot **`yield`** against **`fertiliser`**. So it is not yet useful as a general tool. It would become much more useful if the command allowed the variables to be named in the command line. We would like to type something like:

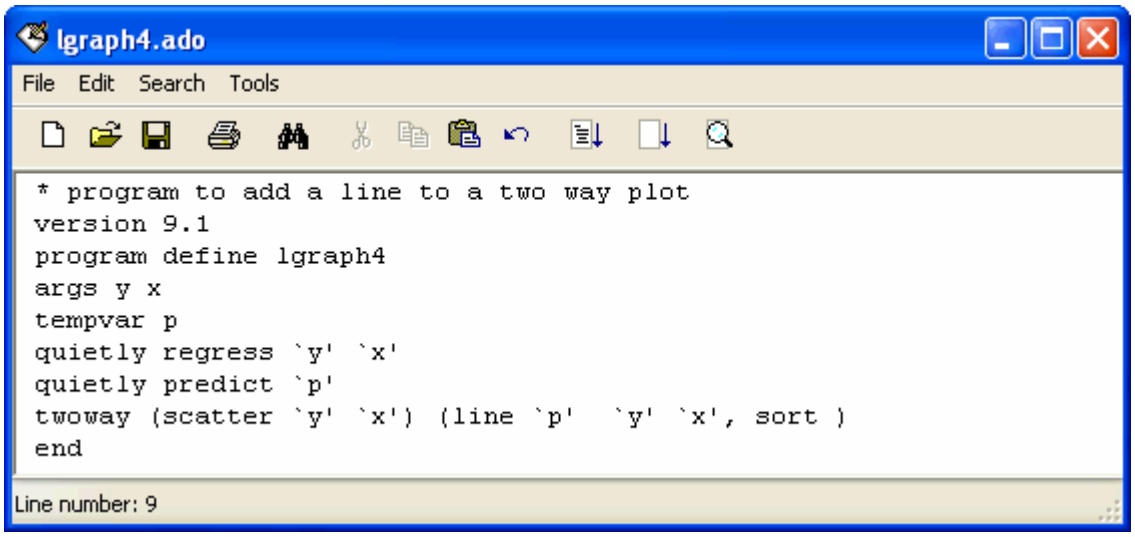
. lgraph3 yield fertiliser

. lgraph3 yield size

and so on.

This is the next improvement, shown in **lgraph4**, see ***Fig. 18.7***.

Fig. 18.7 Making the command more general



The name is changed to **lgraph4** on the third line. Then there is a new 4th line that starts **`args`** (short for arguments). These temporary variables, **`y`** and **`x`**, are used just as we used **p** earlier. As with **p** they have to go in the special quotes that were introduced in Section 5.5.

Now use **Tools ⇒ Do again**, and then test the program by trying a new graph, namely

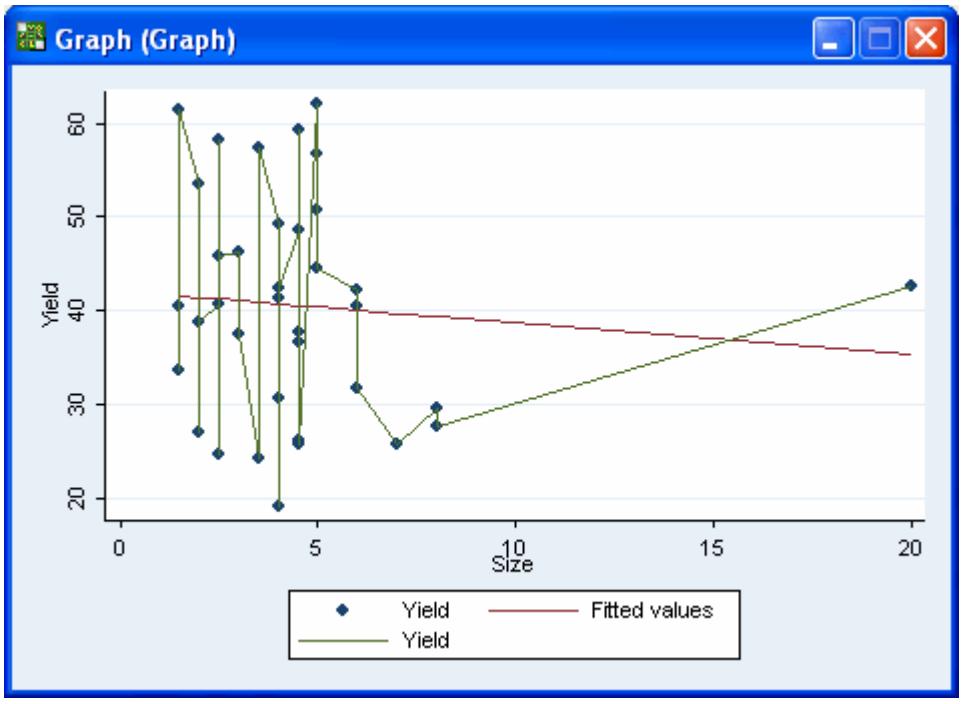
. lgraph4 yield size

If you copied the changes exactly as in ***Fig. 18.7***, then the graph looks odd, see ***Fig. 18.8***. The mistake is the extra `y' in the line part of the twoway command. Correct this mistake, then type:

. program drop lgraph4

Save the file, giving it the name **lgraph4.ado**. Then try the command again. The extra lines should have disappeared.

Fig. 18.8 Results from running lgraph4



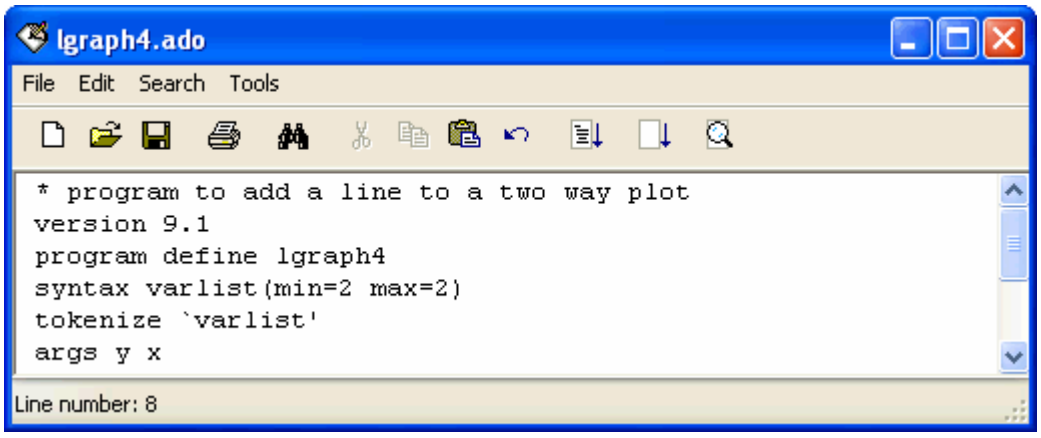
If you run a new command, like **lgraph4** straight from a file, then it first copies the command to memory, and then executes it.

Suppose you then find a mistake in the command that is in the file. You could correct the file in the usual way and then save it, and run it again. But it will NOT run the new version, because it already has a copy of the command in memory. To make it run the new version you must still drop the old command, by typing

. program drop lgraph4

Try this by adding another two lines to the program above, see ***Fig. 18.9***

Fig. 18.9 Checking the syntax with the command



In ***Fig. 18.9*** the syntax command states that lgraph4 expects a list of exactly two variable names (**`min and max both 2`**), and it places them in the local macro, called **`varlist`**. The tokenize command breaks `varlist' into the individual variable names and puts them into local macros 1 and 2 (called tokens). Then the args command copies the contents of these tokens into local macros **y** and **x**. **Use File ⇒ Save** to copy the improved file back with the same name.

To see the advantage of this version, try the command with an error, before deleting the previous version:

. lgraph4 yield size,

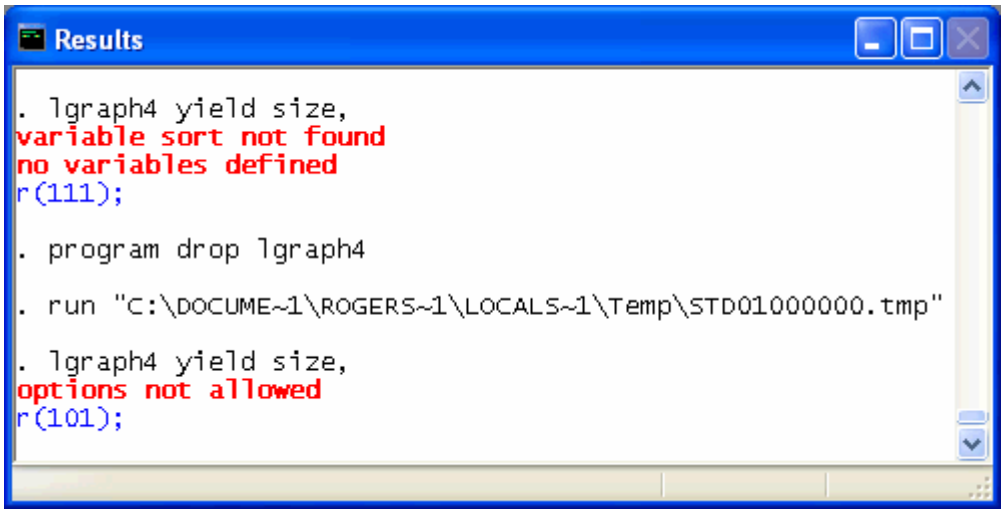
where you have added a comma at the end, as though you will give an option, but have not. Stata responds with an incomprehensible error message, see ***Fig. 18.10***. Now type

. program drop lgraph4

. lgraph4 yield size,

There is now a clear message, see ***Fig. 18.10***, that options are not allowed.

Fig. 18.10 Results when errors are made



18.5 Allowing if, in and options

The command lgraph4 is now sufficiently general to be a useful personal program. To make it more widely available it should also allow for other aspects of the Stata command line, like **if** and **in**.

Part of the power of Stata is the ease with which these aspects can be added.

Edit the file as shown in ***Fig. 18.11*** and name it as **lgraph5**.

In ***Fig. 18.11*** the syntax command states that the **lgraph5** command must be followed by two variables, and that **if** and **in** are optional. The * after the comma, also in square brackets indicates that any options can also be included.

The options to permit **if**' or **in** have also been added to the **regress**, **predict** and **twoway** commands. Finally the **options** have been added to the end of the twoway command, to allow options when plotting the graph.

Fig. 18.11 Adding if and in to the new command

lgraph5.ado

File Edit Search Tools

```
* program to add a line to a two way plot
version 9.1
program define lgraph5
syntax varlist(min=2 max=2) [if] [in] [, *]
tokenize `varlist'
args y x
tempvar p
quietly regress `y' `x' `if' `in'
quietly predict `p' `if' `in'
twoway (scatter `y' `x' `if' `in') (line `p' `x', sort ), `options'
end
```

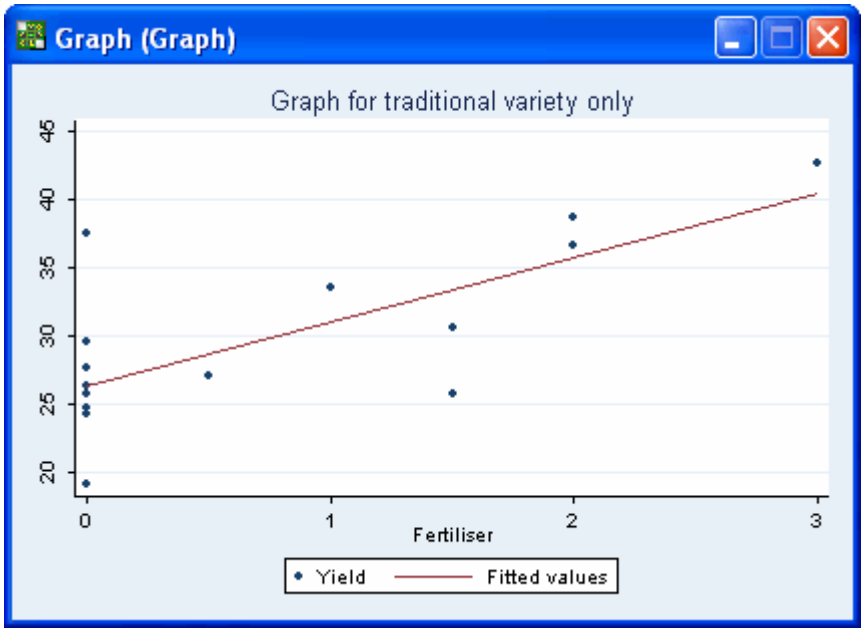
Line number: 11

So now you could give the command as

. lgraph5 yield fertiliser if variety ==”TRAD”, title(Graph for traditional variety only)

The result is shown in ***Fig. 18.12.***

Fig. 18.12 Graph showing the use of an option to give a title



Of course you could give any option to **lgraph5**, but Stata will give an error message unless it is valid for the **twoway** command.

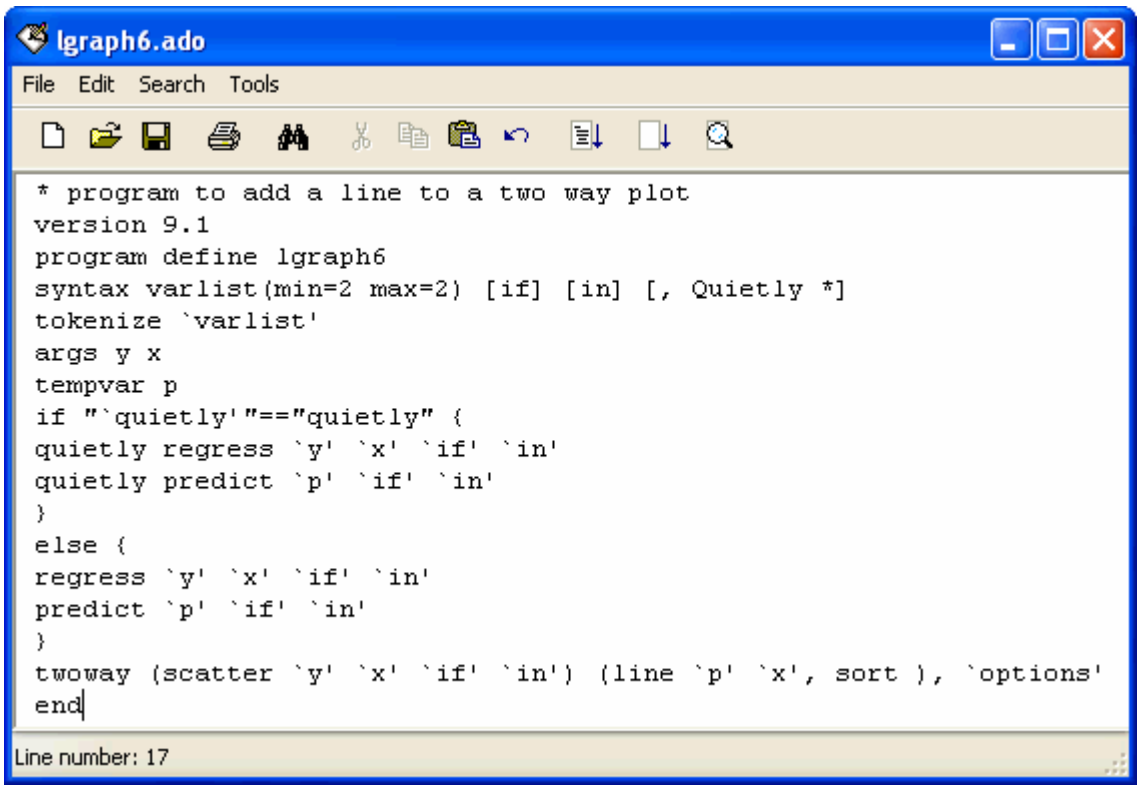
18.6 Adding flexibility to the command

The final improvement to the command involves adding another option. This time it is a specific name of our own choosing.

We argued earlier that if all we want is the fitted line, then we can avoid having the output from the regression command. That is why the prefix, quietly, was added in front of the regress and

predict commands. The result is now similar to the output when **Graphics ⇒ Easy graphs ⇒ Regression fit** Is used.

Fig. 18.13 Adding a new option to the command



Suppose sometimes the regression output is needed, and on other occasions, just the graph. An option is added that we have chosen to call **Quietly**, when giving the syntax, see ***Fig. 18.13***. Then a conditional part of the code is added, to execute some lines if the option quietly has been set, and other lines if it has not.

Also change the name to **lgraph6**, and save the file as **lgraph6.ado**.

Now if you type the command as

. lgraph6 yield fertiliser

you should get the regression output as well as the graph. Typing

. **lgraph6 yield fertiliser, quietly**

should just give the graph. In the syntax line just the **Q** of **Quietly** was given as a capital letter. This is then the minimum abbreviation, so

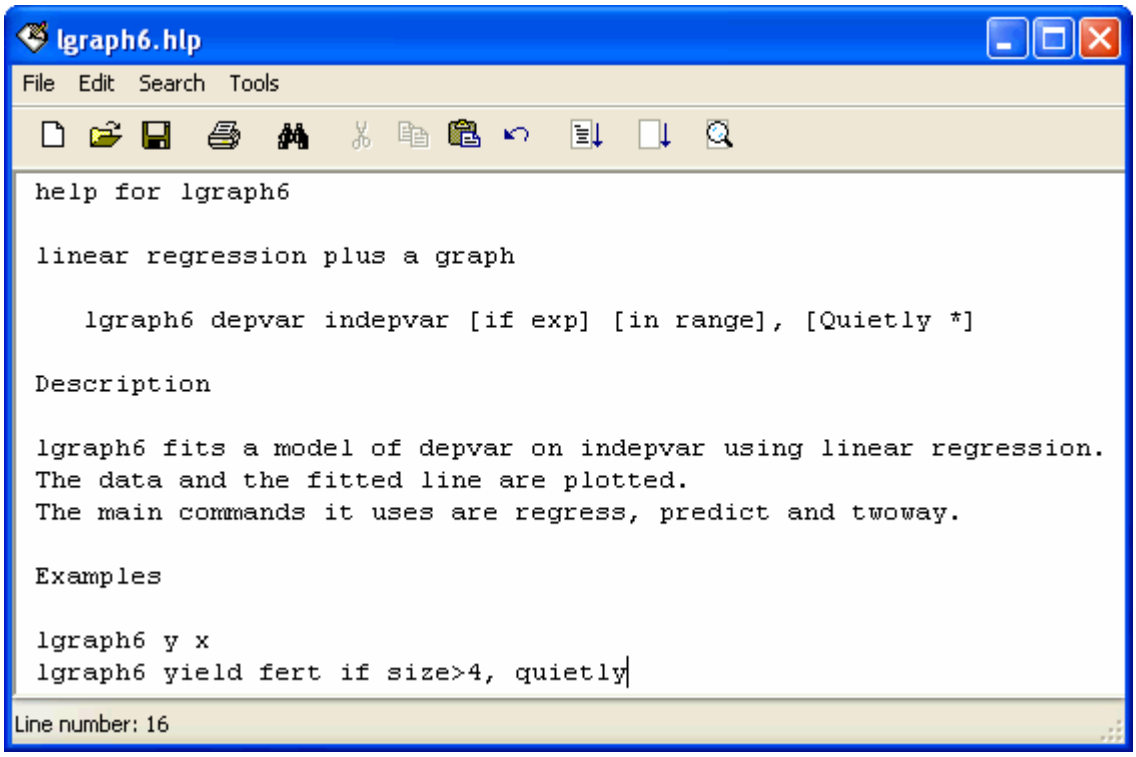
. lgraph6 yield fertiliser, q

could also be given.

18.7 Adding a help file

Now you have a working program that could be distributed in your organisation. But you also need to distribute information on how the command can be used. An easy way is to add a help file, as described in Chapter 17.

Fig. 18.14 Adding a help file to the run command



With Stata you can write the help in a simple text file. Then save it with the same name as the command, but the extension **hlp**. You can prepare the file in any editor and ***Fig. 18.14*** shows an example where we have just used Stata's usual do-file editor. When using **File ⇒ Save As,** make sure to change the extension to hlp.

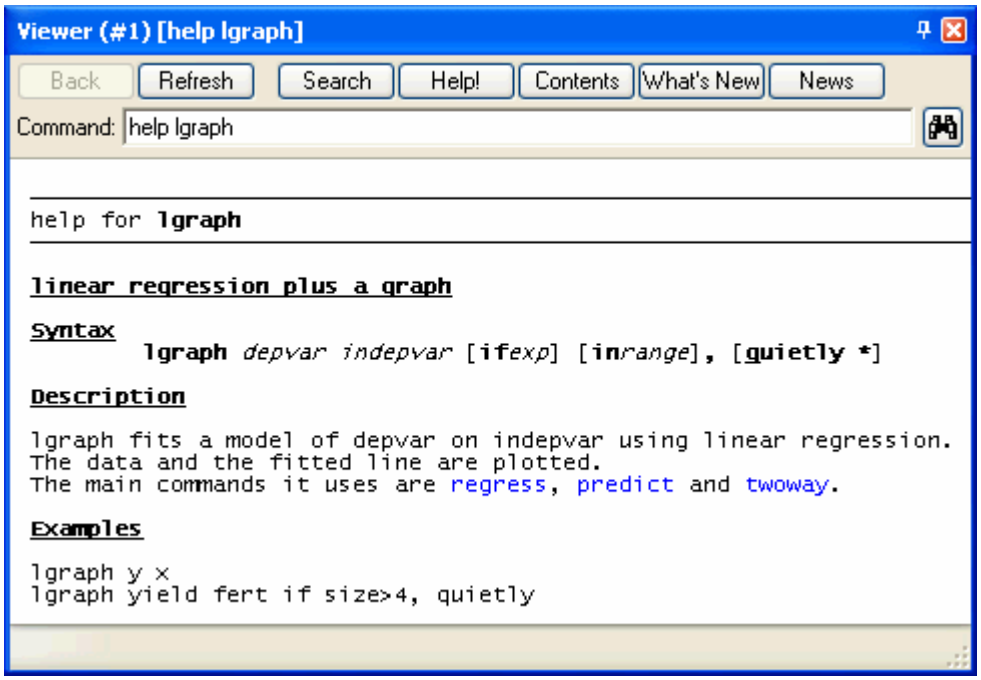
Then try the help by giving the command

. help lgraph6

The text should now appear in the Stata viewer.

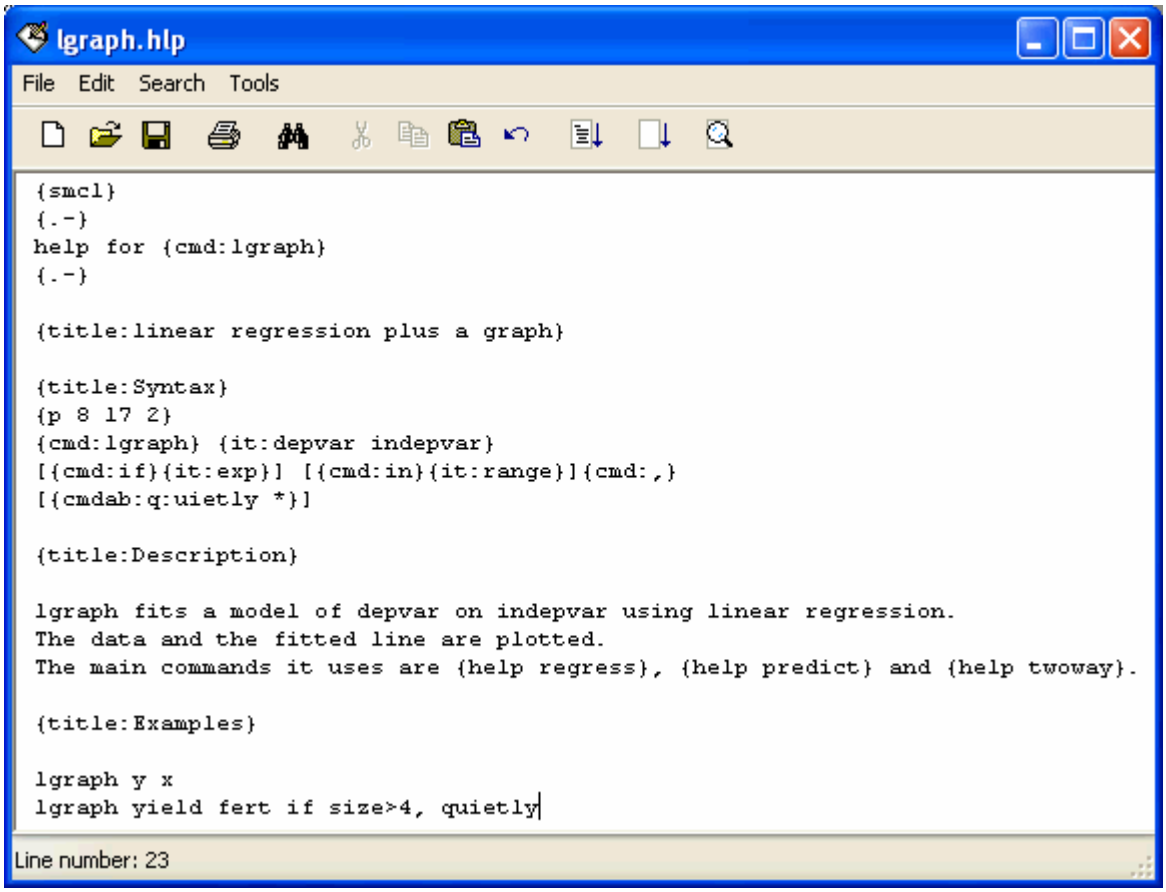
Fig. 18.15 shows the text from ***Fig. 18.14***, but displayed in roughly the same way as other Stata help.

Fig. 18.15 Formatted help for the run command



The file for this is shown in ***Fig. 18.16***. The explanations of the features enclosed in { } were given in Section 17.2.

Fig. 18.16 File to give the formatted help



The command and the help are renamed as **lgraph**. They are both supplied on the distribution CD.

18.8 Making a dialogue

A dialogue can be added so the new command is easier to run. This is shown in ***Fig. 18.17*** and can be called up in the usual way, by typing the command:

. db lgraph

Alternatively it can be added to the User menu, as described in Chapter 17. It looks just like a standard Stata dialogue. Try it in a variety of ways. The help button should work, and bring up the help file, shown earlier in ***Fig. 18.15***. Run as shown in ***Fig. 18.17*** and you generate the command

. lgraph yield fertiliser

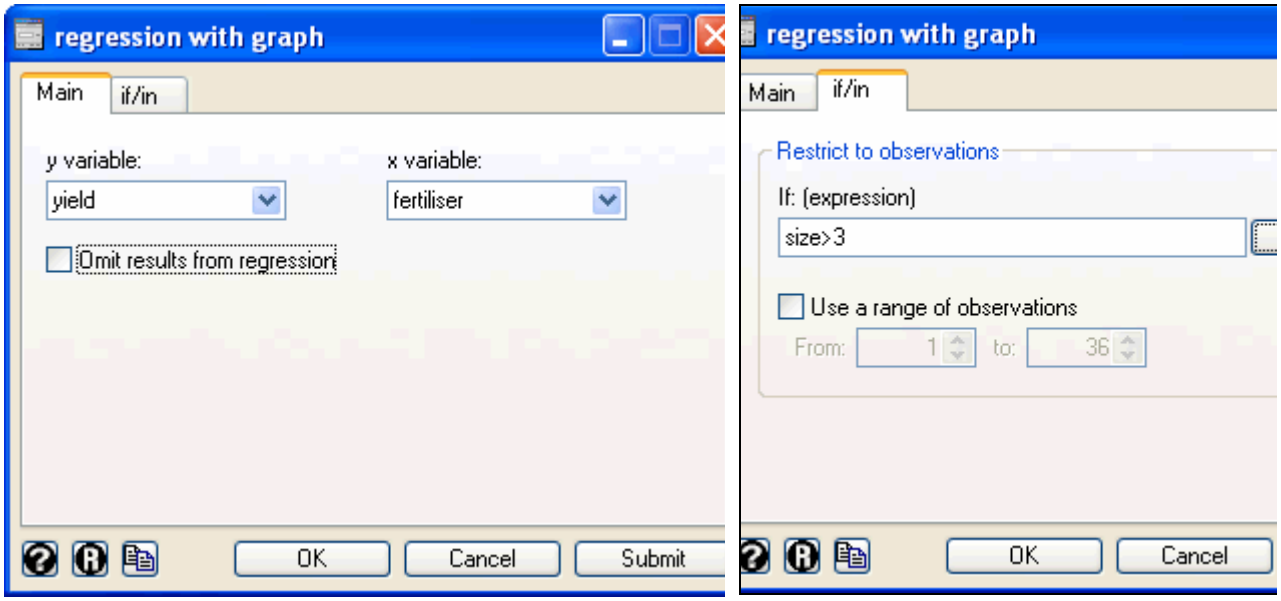
If you tick the box labelled “Omit results from regression”, then it gives

. lgraph yield fertiliser, quietly

Fig. 18.17 also has a tab for the **if** or **in** options. Try with the condition “if (size>3) to look at the graph for only the large fields. This corresponds to the command

. lgraph yield fertiliser if (size>3)

Fig. 18.17 Adding a dialogue for the new command



This dialogue results from yet a third file that you need to program. We already have **lgraph.ado** with the command and **lgraph.hlp** with the help information. Now you need to write a file called **lgraph.dlg**. The code is shown in ***Fig. 18.18***.

If reading this chapter was your first experience in programming, then you might feel that we are attempting the impossible to show you the commands in ***Fig. 18.18*** that add a dialogue to the **lgraph** command. But reflect first on your objectives from this chapter. If you are not a programmer yourself, then our aim is for you to understand what is possible, rather than teach programming.

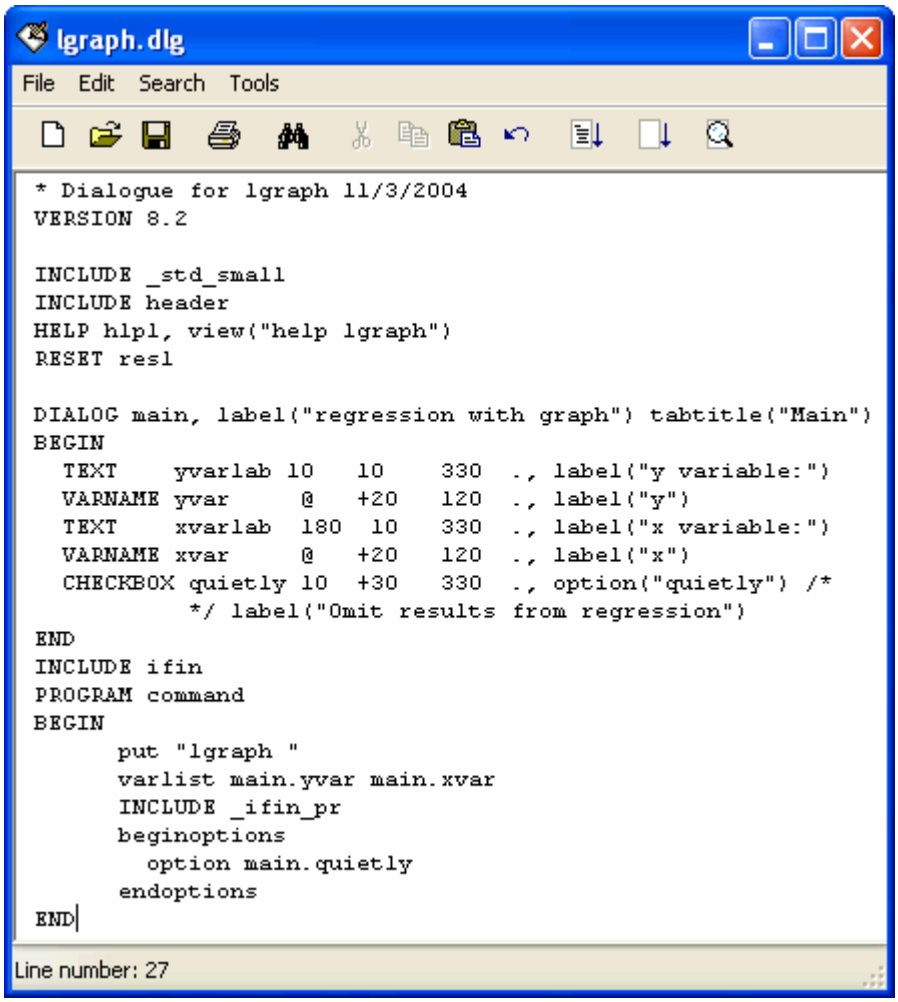
It is impressive that a small amount of code in ***Fig. 18.18*** has produced such a neat-looking dialogue, ***Fig. 18.17***, just like the ordinary Stata dialogues introduced in Chapter 1. ***Fig. 18.18*** shows that it does not take long for someone with experience to add a dialogue to an existing command.

For those who wish to learn more about the code itself, some of the components of ***Fig. 18.18*** are explained briefly.

There are four lines that start with the **INCLUDE** command. They each call standard dialogue files that the Stata developers have written, that are used to construct other dialogues. The line **INCLUDE _std_small** includes the code to make a small dialogue of standard type. Then the command **INCLUDE header**, adds the standard **OK, Cancel** and **Submit** buttons, see ***Fig. 18.17***. The next two lines add the standard **HELP** and **RESET** buttons. The **HELP** command also states which help file is to be activated, if that button is pressed.

The part of the code between **BEGIN** and **END** provides the information on the dialogue seen in ***Fig. 18.17***. There are 5 elements there, namely two bits of text, two boxes into which the variables are entered, and one check-box.

Fig. 18.18 Program to make a new Stata dialogue



The line **INCLUDE ifin**, is very good value. It adds the standard tab, see ***Fig. 18.17***, so you can include this feature when using the command.

The last part of the code in ***Fig. 18.18*** collects the information from the dialogue, and constructs the command.

At the top of the code in ***Fig. 18.18*** there is the line **VERSION 8.2**. The code was first written for the previous version of Stata, when dialogues were different. They did not have the pull-down feature for choosing the variables, nor the possibility of copying the command to the clipboard. The same code works in Stata 9, and, as shown in ***Fig. 18.17***, these new features are added automatically.

18.9 Adding to the menu

Finally it would be good if you didn't have to type

. db lgraph

or

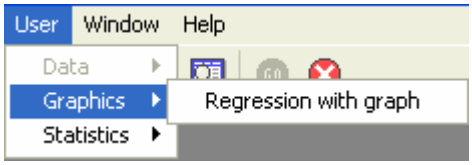
. db lgraph6

to get the dialogue. Section 17.1 showed how to add to the menu system. The ideas are briefly reviewed here.

In the command window type:

. window menu append item "stUserGraphics" "Regression with graph" " db lgraph"

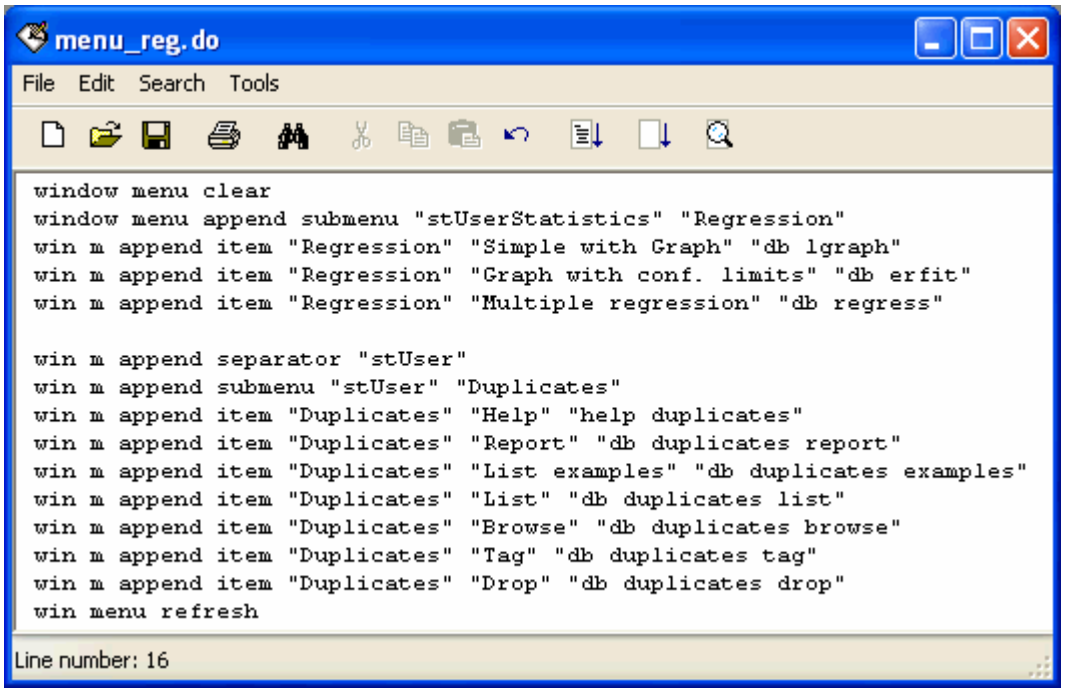
Fig. 18.19 Adding the regression command to the menu



Then, when you use the User menu, see ***Fig. 18.19***, there is an item under **Graphics**. Click on this item to give the dialogue shown in ***Fig. 18.17***.

This was a long command to type, so you would usually put it in a **Do file**. In ***Fig. 18.20*** a regression section has been added to the code shown in ***Fig. 17.5.***

Fig. 18.20 Commands to add regression facilities to the menu



The user menu is now as shown in ***Fig. 18.21*** and ***18.22***. The first commands in ***Fig. 18.20*** made a submenu of the **User ⇒ Statistics menu**, called **Regression**. Under this, there are three items, consisting of the new command, the one used in Chapter 12 to plot confidence limits, and the ordinary regression menu. The menu is shown in ***Fig. 18.21.***

Fig. 18.21 New regression menu

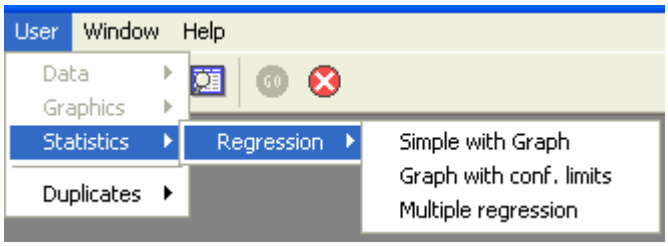
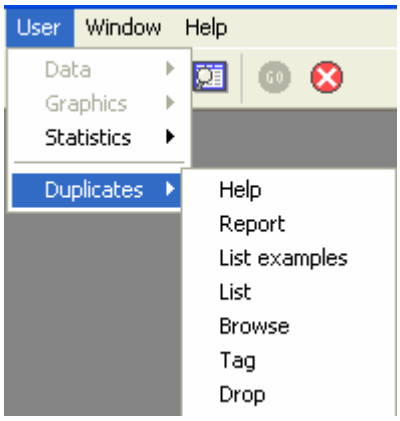


Fig.18.22 Duplicate menu also



The menu on duplicates, described in Section 17.1 has also been included.

18.10 In conclusion

This chapter has shown that it is relatively easy to add new features to Stata, and also to add the extra components of a help file, dialogue and menu, so the additions can be used in just the same way as any of the existing Stata facilities.

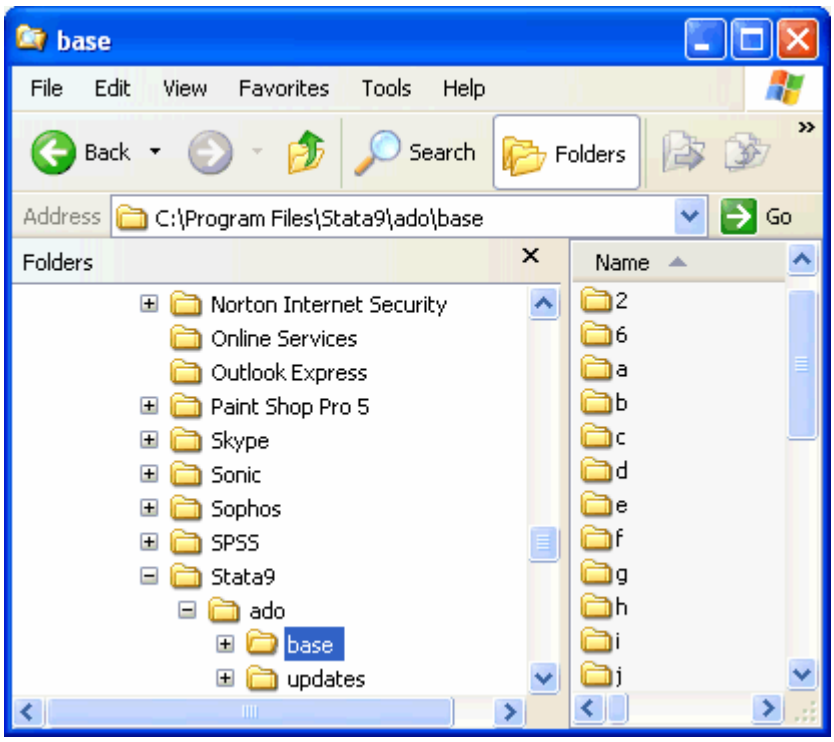
Chapter 19 How Stata is organised

This chapter describes the structure of Stata. It shows how to update Stata over the internet, or locally, how to install commands contributed by users and how to use the Stata FAQs.

19.1 The structure of Stata

Stata does not come as a huge monolithic program that the user is unable to modify. Instead the philosophy is to allow the user as much control as possible. There is a relatively small compiled file that carries out the task of organising and interpreting the rest of the software, including the data input. This file, called **wstata.exe**, for our version of Stata, was roughly 6 mbytes, which is very small by modern standards.

Fig. 19.1 Structure of Stata's files



Most of Stata comes as independent files to which the user can gain access. These are called **ado** files, which stands for **automatically loaded do files**. They have the extension **ado**, so for example the program code for the **codebook** command is in the file **codebook.ado**. There are many hundreds of ado files, and as indicated in ***Fig. 19.1***, they are installed in a subdirectory of the Stata9 directory. Because there are so many files, they are each put in a directory that corresponds to the first letter of the command.

Many of these commands were written by users, and adopted by the Stata Corporation after checking.

As shown in Chapter 18, each ado file itself consists of Stata commands. These files now often come in threes, see ***Fig. 19.2***. For the **codebook** command there is **codebook.ado**, then there is **codebook.dlg**, which gives the dialogue, and **codebook.hlp** that gives the help information. So when you type

. codebook

then Stata loads and runs the file **codebook.ado**. If you type

. db codebook

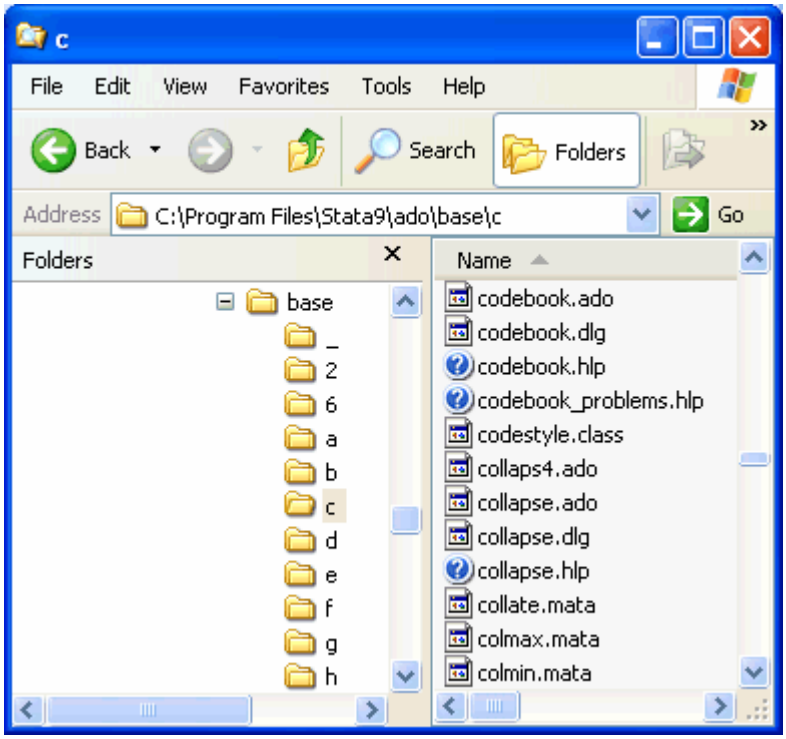
then Stata runs the file **codebook.dlg**, which displays the dialogue. And typing

. help codebook

is an instruction to Stata to load and display the file **codebook.hlp**.

In ***Fig. 19.2*** Windows explorer has indicated that the file **codebook.hlp** is a Windows-style help file. It makes this assumption, just because the extension to the filename is **hlp**. It is not a Windows help file, as you will be told if you click on it. Instead each of these three files are simple ASCII files that you can examine in Notepad, or using the **do** file editor in Stata.

Fig. 19.2 Some of the ado, hlp and dlg files



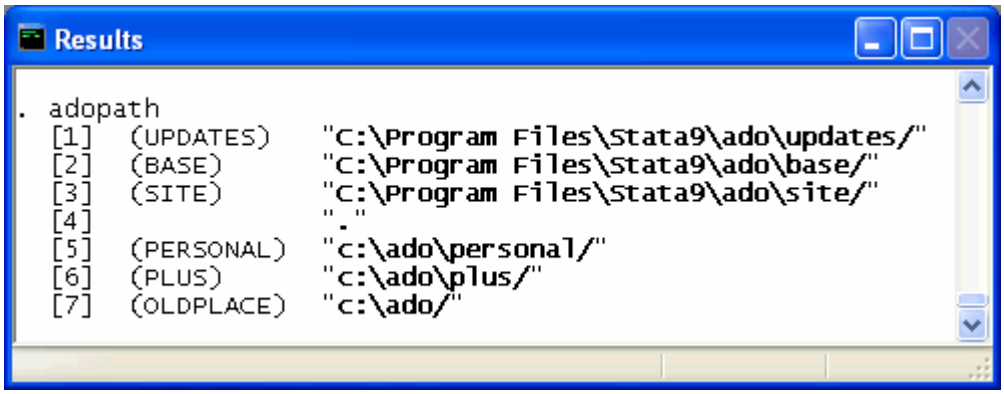
So, Stata is a very open system. Although few will want to change the standard commands, you have access to the code, and so could make changes if you wish. What is likely is that users or organisations may wish to add trilogies of their own, as in Chapter 18, when we added **lgraph.ado, lgraph,hlp** and **lgraph.dlg.**

An important command to understand how Stata is organised is **adopath**. Try

. adopath

The result we found is shown in ***Fig. 19.3***. Note that Stata accepts either forward slash or a backward slash in path names.

Fig. 19.3 Directions used by Stata to find comands



What you find on your machine will depend on where Stata was installed. If it is on a network server, then the first three directories might have a drive letter, for example N: instead of C:\Program Files\.

The paths in ***Fig. 19.3*** are listed in the order in which they are searched. For example, to find **codebook.ado**, Stata first looks in the UPDATES directory, to see whether the original **codebook.ado** has been updated. If it is not there it looks in the BASE directory, and so on, down the list.

Stata ignores directories that are non-existent. For example, on our machine there was no SITE directory. But the availability of this directory shows the potential for a site using Stata to produce extra commands and make them available to everyone. They just have to be copied to the correct directory, perhaps one that is shared over a network, or copied to each individual machine.

The fourth entry in ***Fig. 19.3***, "**.**" stands for your current working directory. This was where Stata looked to pick up the file for the commands in Chapter 18.

The rest of the list is to help you in customising Stata. For example you may have some personal commands that you choose to store in **C:\ado\personal/** or you may have downloaded some commands from the internet, or been sent extra commands that you have installed in **C:\ado\plus/.**

Additional paths can be added to the search list, as in

. adopath + C:\courses\ado/

Similarly paths can be removed, most easily by number. For example:

. adopath –3

will remove the SITE directory and re-number the rest. It is sometimes useful to add a path to the start of the search list. Try

. adopath ++ C:\courses\ado

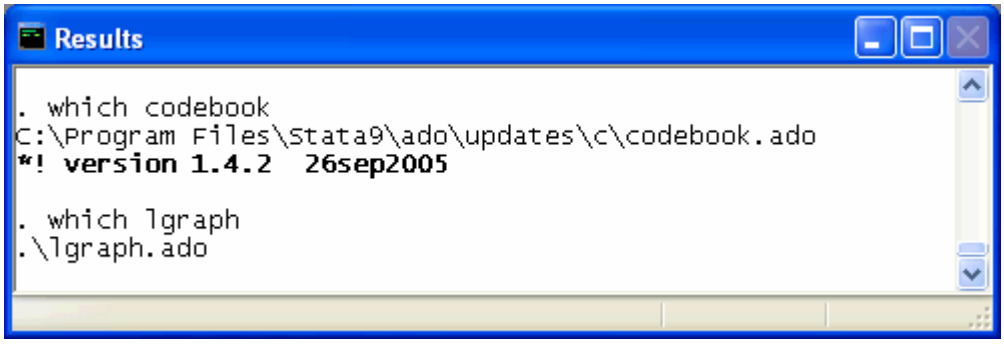
to add **C:\courses\ado/** to the start of the search list. The main reason for doing this would be if you have altered some of the standard commands in Stata, and would like your own version to be used. You should not change the version in the UPDATES or BASE directories, because any changes here may be destroyed, when you next update Stata. Instead copy your improved version to a different directory, and instruct Stata to use that version.

To find from which directory a particular command has been used, type

. which codebook

The results, for us, are in ***Fig. 19.4***. Similarly we found where **lgraph** (from Chapter 18) was called.

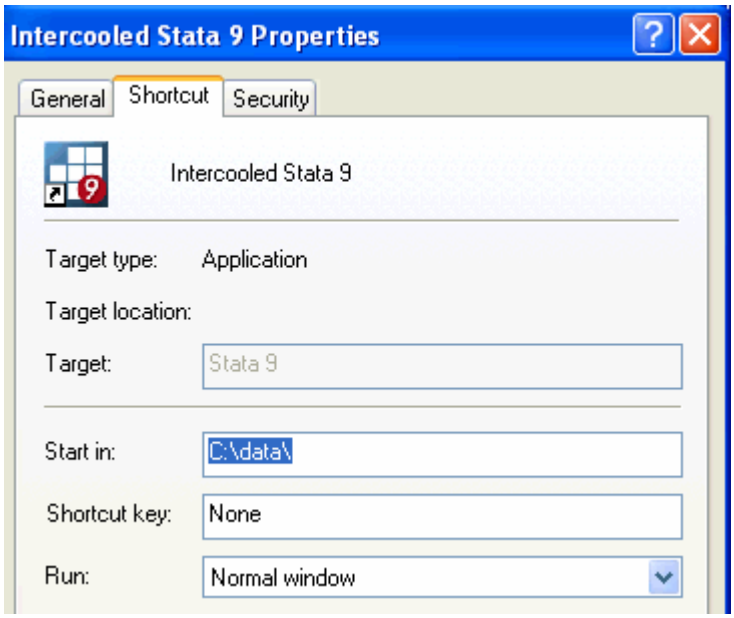
Fig. 19.4 The which command to locate an ado file



19.2 Starting Stata

When you start Stata, you are probably running from a short-cut on your desktop. If you right-click on the icon and then choose the Shoertcut tab, you will get a menu roughly as shown in ***Fig. 19.5***.

Fig. 19.5 Tailoring how you start Stata



This would be one way to change the starting folder to something more appropriate than C:\DATA.

If you leave the starting folder as C:\DATA, then when you start Stata you can type the command

. **cd**

This will inform you that C:\DATA is the current folder. In ***Fig. 19.3***, which shows the results from the command

. **adopath**

the 4th directory was labelled ".". This also corresponds to this folder, C:\DATA. You can always use the cd command to change this folder.

When you start Stata, it looks for an initial file called **profile.do**. If it finds this file, then it runs it, before handing control to you. That is a way of changing the initial memory for Stata, for example by making this file with the command:

. set memory 10m

You may also wish to open a file to log commands, in **profile.do** as in

. cmdlog using `c:\temp\cmdlog.txt`**, append**

The **append** option here, keeps the command log from previous sessions, so you can examine past commands.

19.3 Updating Stata

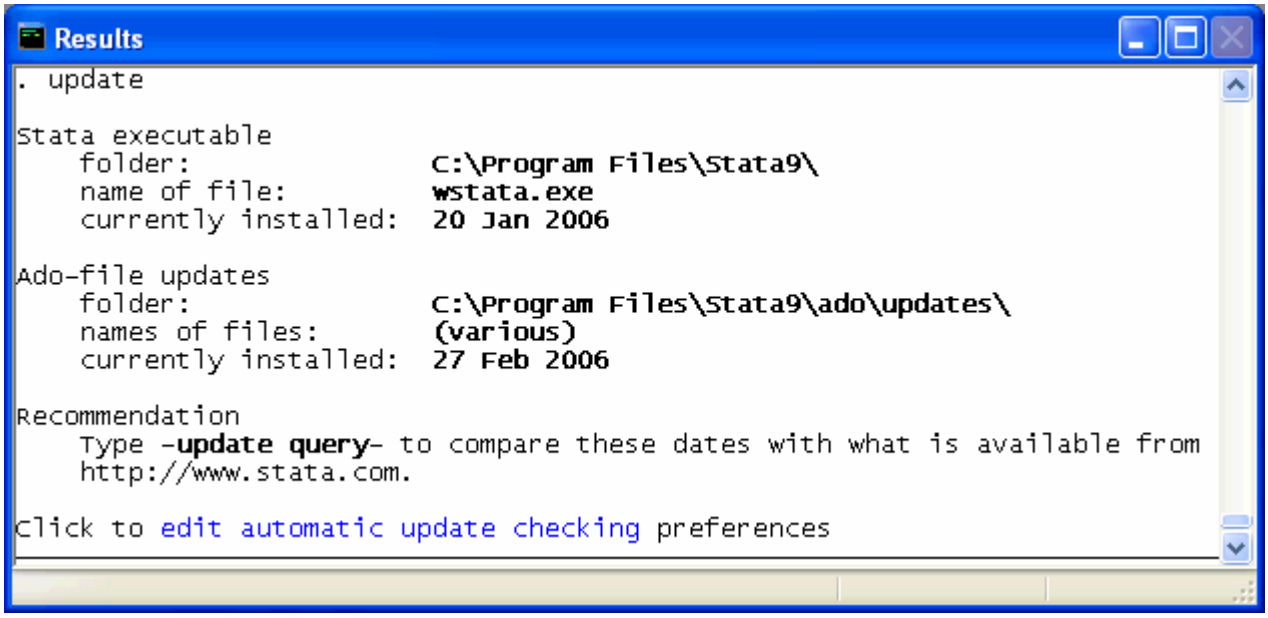
The way Stata is organised makes it important to update the package regularly. How you do this depends on whether you have a direct connection to the internet, or use Stata over a network, or perhaps on a stand-alone machine.

It is very easy if you have an internet connection, and in any case you will use the update command. Start in Stata by typing:

. update

For us this gave the summary shown in ***Fig. 19.6.***

Fig. 19.6 Update command reports on the current version of Stata on your machine



If you follow their recommendation, in ***Fig. 19.6***, then (do not do this yet) you type

. update query

This will connect you to www.stata.com, where your version is checked against the current executable and ado files.

If this works, then you will get a report of your system, and advice on whether it needs updating. The advice may be to update the executable (the Stata core program), or the ado files, or both. In response to the advice type one of

. update exec
. update ado
. update all

A typical update may take 15 minutes on a reasonable internet connection. Stata write confidently that there will be no problems if the connection goes down during the copying, and you need to restart the procedure on a later date.

If you cannot connect to www.stata.com, but would like to connect directly in the future, then open the **General Preference** dialogue box.

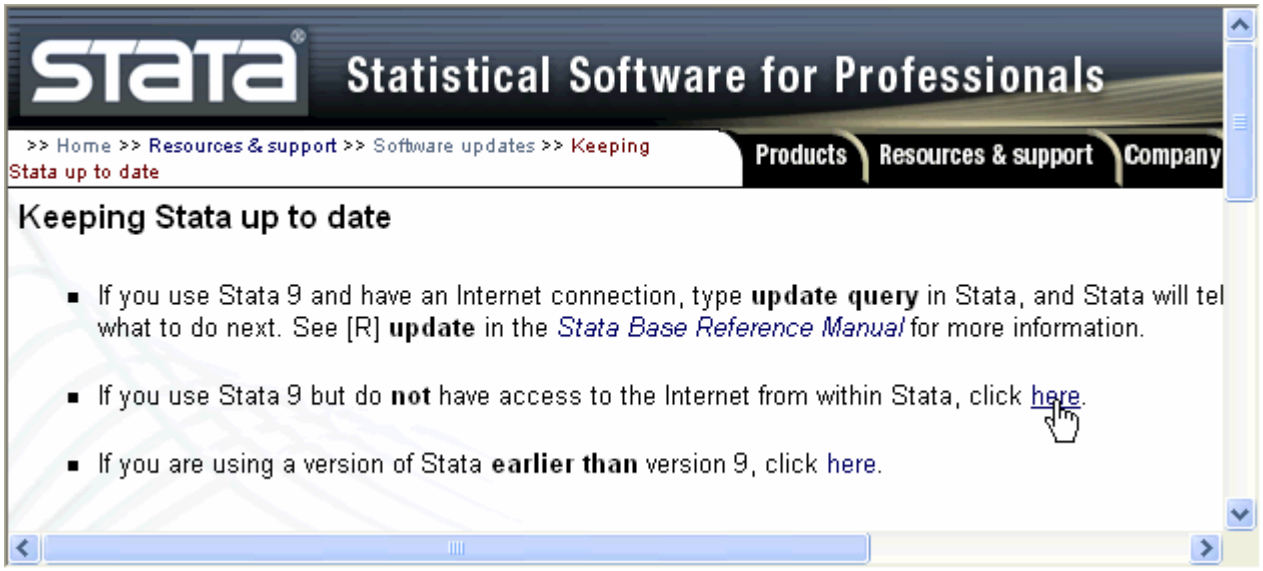
Go to **Prefs** ⇒ **General Preferences** ⇒ **Internet Prefs** tab and fill in each text box, finding the details from your internet administrator. Then try the **update query** command again.

If it still does not work, you need to update in the same way as on machines with no direct access to internet.

You need to find a machine with an internet connection. Then go directly to the Stata site, see ***Fig. 19.7.***

Fig. 19.7 The main Stata page on www.stata.com and the page for updates

Fig. 19.7 continued: the first update page



Choose the option for Resources and support, and then Software updates. Then click on the appropriate option in the following screen, ***Fig. 19.7***. Alternatively, go straight to www.stata.com/support/updates/stata8.html

This page will provide instructions on how to copy and then install the updated exec file and the ado files. The information also shows the dates that these files were last changed, see ***Fig. 19.8*** for an example. So the dates can be compared with the results from the update command on your machine, ***Fig. 19.6***, to see if a more recent version is available.

Fig. 19.8 Information from Stata on the most recent version

If you copy both the exe and the ado files, the first stage in the procedure we followed was as follows.

- Copy the **wstata.bin** file into the Stata program directory.
  Rename the previous **exe** file, which for our version was called **wstata.exe**, into **wstata.old**.
  Rename the **wstata.bin** file we had just copied into **wstata.exe**.

That is all you need to do to update the exe file. The ado files are changed more often, so you may do this second stage, without needing to update the exe file.

Unzip the ado update file into a temporary directory.

Go into Stata. You can check that the exe file has been updated by typing **update** again.

Type the command such as:
. update from "c:\temp\stata"
Choose the directory where you unzipped the ado.zip file.

If you have a site licence, and are updating over a network, then just give the network directory with the files instead.

If the command works, then you will see on the screen that it is copying lots of files. (For us, it worked almost every time. The only time it failed, was when it said we were up-to-date already, when this was clearly untrue. On closer investigation, the zip file

had not been copied over correctly. We therefore downloaded again, and the updating worked ok.)

19.4 Adding user contributed commands

User-contributed commands are supplied without any guarantee that they will work, but they are usually of a high standard. One such command was used in Chapter 11, because Stata does not yet have any built-in facilities for tabulating multiple responses.

In Chapter 18 we prepared our own command, to show how user commands can be written.

Providers of commands usually make them available in what Stata calls a package. This consists of the files themselves, plus some index files, so Stata can recognise which files to install. Just as when Stata itself is updated, these files can be installed directly from the internet, if you have web access, or they can be downloaded to a CD, or directory, and installed from there.

Once you know the name of a package, then Stata's net command is used to handle the installation. For example, on the CD with this book we described all the files to install in a special file, called **SSCStata9Guide.pkg**. Once you know this name, then use

. net from D:

to move to the drive or directory where the package is available. Then

. net install SSCStata9Guide

to install the program and help files. Then

. net get SSCStata9Guide

to add all the data and other ancillary files. In this case, the **net install** and **net get** commands as above, they may not to have worked because the files were already there from earlier, see Chapter 0. If you add the replace option, i.e.

. net install SSCStata9Guide, replace

. net get SSCStata9Guide, replace

then this should work.

If you would like to make the extra commands and sample datasets into a package that others can use, see the help in Stata, typing

. help usersite

. help ssc

Here, SSC stands for the Statistical Software Components (SSC) archive, provided by http://www.repec.org .

19.5 Support when using Stata

Apart from the Help, the main sources of information about Stata are the User Guide and the Reference Manuals. Stata also has a special manual on the commands for processing surveys. In addition Stata Corporation and the agents in different countries offer internet-based courses.

Stata has a technical support group that will sort out any problems for registered users, but before contacting them you are advised to check the FAQs and other sources of documentation. See the Stata web page, under User Support and Technical Support for more details.

A number of useful books have been published for learning more about statistics while using Stata. See the Stata webpage under Bookstore for more details.

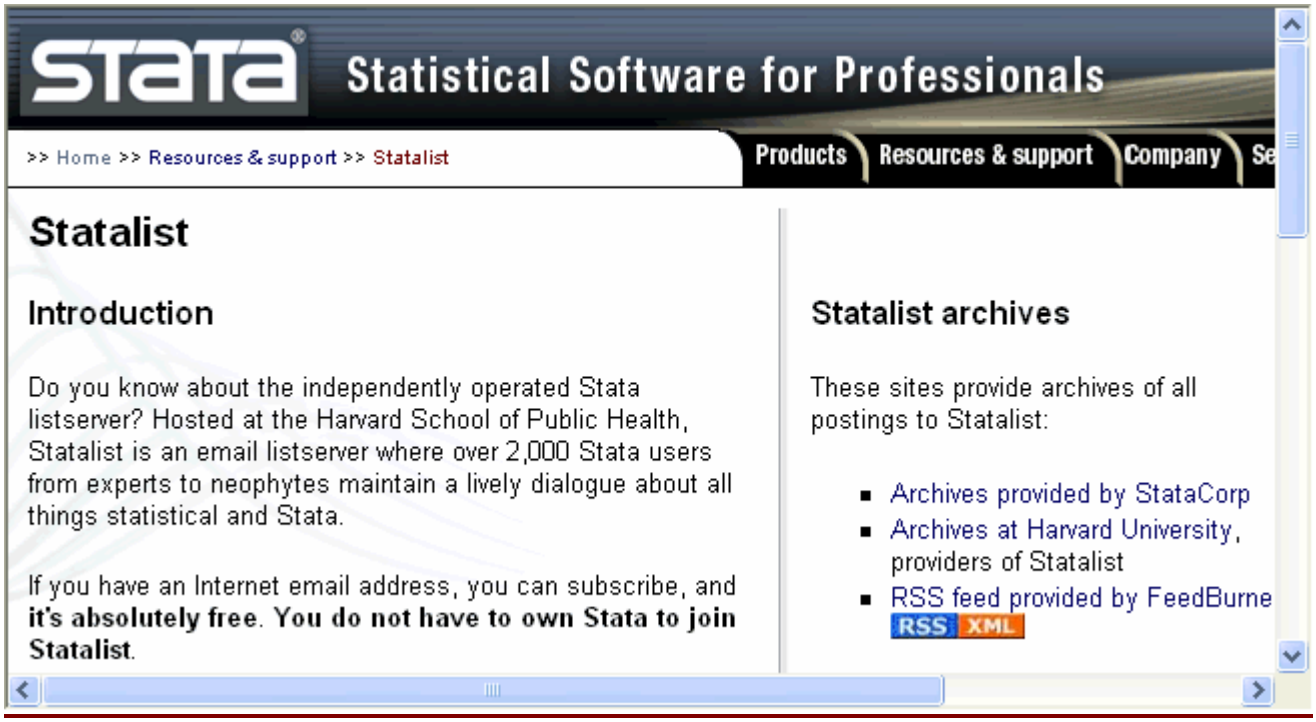
The Statalist is a useful resource for both beginners and experienced users of Stata. This is a listserver that distributes messages to all subscribers, and subscription is free. This is independent of Stata, though the also look in, for problems with the current version of the software and suggestions for the next release.

To join the list, follow the instructions given on the Stata site, www.stata.com/support/statalist/. See ***Fig. 19.9*** for further information. There is even a digest version of the list, which may be needed for those who have slow e-mail access.

The list is mainly to share information, rather than as a resource for help in the use of Stata. The Stata community is generous with its help. You can ask for help over the list, but first check the manuals and the FAQs at the Stata website.

There is also a Stata journal that you can subscribe to. This is not free, but is modestly priced. Information is on the Stata web site, including instructions for authors. Abstracts of papers can be viewed without subscribing, and any **ado** files are available freely.

Fig. 19.9 Information about the Stata list



Chapter 20 Your verdict

One reason for writing this guide was to help those who would like to evaluate Stata as a statistics package for the analysis of their survey data.

The examples are mainly from Africa, and this is because the first organisations who are using this guide to help in their evaluation are the Central Bureau of Statistics, (CBS) based in Nairobi, and the Uganda Bureau of Statistics (UBOS). In this section we give our opinion of Stata, having written the guide, plus the views of CBS and UBOS staff, following initial courses.

At a simple level, a "verdict" would be either to adopt Stata, or to use an alternative.package. For individuals the decision might be this simple, but organisations can have more general solutions to satisfy their needs. For example they might decide on a strategy of continuing with a spreadsheet for most people, but suggesting a statistics package for some headquarters staff, and here allowing staff to choose between SPSS and Stata on an individual basis.

In giving our verdict we do not attempt to compare Stata with other software directly. Such comparisons need to be made by the individuals concerned and they change quickly as the different statistics packages advance. For example until 2003 a major limitation of SPSS was that it had no special facilities for calculating standard errors for surveys (the material described in Chapters 14 and 16). This is available from SPSS Version 12.

Instead, we describe what we consider to be strengths and weaknesses of Stata, for processing survey data. It is then up to the reader to assess whether other packages are more appropriate, or perhaps that we are not using all the key criteria.

20.1 Getting Stata and keeping it up-to-date

Stata is not free software, but it is very reasonably priced. In addition, the suppliers are prepared to allow government organisations in developing countries, to be provided with Stata at the same (educational) price as the local University. At least this was allowed for Kenya and Uganda. This aspect reflects the fact that government agencies in many developing are partially dependent on donors for buying and upgrading software.

It is useful that Stata is bought, rather than leased. We initially bought a perpetual licence for Version 8.0 for CBS in early 2003. This was updated to Version 8.2, in early 2004, by downloading files from the suppliers as described in Chapter 19. Each version is available for a number of years. When, Version 9 was produced, it had to be bought. But if funds are not available staff can still continue with their analyses using Version 8, which continues to be supported. This is not the case with software that is leased on an annual basis.

The cost of updating is less than the cost of buying from scratch. For UBOS, Version 8 was bought in 2005, just prior to the release of version 9. The cost of updating the site licence, including new guides, was very modest, (less than the cost of a new single-user licence).

We also like the fact that Stata comes (more or less) complete. We do not have to make decisions on whether particular components can be justified. One exception is StatTransfer if files are to be imported or exported between packages, see Chapter 3.

Stata was provided with all the printed manuals. And delivery was excellent. Within one week of deciding on the purchase, the software and manuals were delivered to Nairobi.

High on our wish list is for the manuals also to be available as pdf files, within the help system. We had multiple licences licences of the software, but only a single copy of the printed manuals and were continually scouring the buildings for a particular guide.

We very much appreciate the support received from enthusiasts we contacted. In addition to the Stata developers themselves, there is clearly an active group of users who help others and provide new features. For example, the lack of facilities for processing multiple response data,

see Chapter 11, was one potential failing. This was resolved by an **ado** file provided by two users, who also responded immediately to queries about possible further features.

20.2 Improvements in versions 8 and 9

Two main developments from Version 7 to Versions 8 and 9 were the new graphics, and the system of using menus and dialogues. The graphics are very impressive; see Chapters 6 and 8.

The production of the graphs lacks the interactivity that other software provides. But for the graphs from large surveys this is outweighed by the value of having the command files associated with the finished graphs. Hence they can be reproduced or the scheme changed with ease.

Many graphics packages provide a very wide range of pseudo-three dimensional graphs and this is thankfully absent from the Stata system. Instead there is a comprehensive guide and system for the types of graphs that are needed. The facilities include combining multiple graphs on a single frame.

Our views on the menus and dialogues are more mixed. Initially they are not as intuitive as some other packages. Each menu corresponds to a Stata command, so when there is a mix of overlapping commands for a given task, then there is now a similar mix of overlapping menus and dialogues. The help on the menus is comprehensive, but also rudimentary. It merely provides the help on the associated command.

The limitations of the current menus were not a particularly serious problem for us. The analysis of large surveys requires users to understand something about the commands, for the reasons given in Chapters 2 and 5. If the menus are viewed as a simple way that users can start their analyses, then they do provide this gentle route. They also generate usable commands and so help in the production of the **do** files described in Chapter 5 etc.

Once we became more used to Stata's menu system the consistency of the structure of the dialogues is excellent. Then the ease with which users can add their own dialogues and menus, as described in Chapters 17 and 18, is particularly impressive.

Version 8 also added ODBC as a way of importing data. Stata is limited in reading directly from other software, and is the only standard statistics package that we use, that cannot read spreadsheet files directly. Getting the Stat-transfer program can solve this, as described in Chapter 3. However, a powerful ODBC facility is useful for survey data processing. The lack of a menu system for the ODBC is a disappointment. However, Version 9 added XML menus for data import and export, which may solve the data transfer problems.

20.3 General points

If you use a spreadsheet for data processing, then you keep everything in a single workbook. With Stata you will have many files, each with a simple structure. Even just the data will be in a range of different files, see for example Chapter 12, where each use of the **contract**, or **collapse** commands produces another file, with summary values. Graphs are in individual files, as are the **do** files. So you probably need to use a different directory for each survey. Some statistical software allows all files associated with a project to be collected together, but this feature is absent in Stata.

Windows users will initially find that some standard features are absent from Stata. For example there is no button or option to undo the past few commands, at least not on the main menu.

Set against this is a considerable "comfort factor" for those organisations who wonder if they might later move from Windows to Unix, or perhaps be provided with some Macintosh computers. Stata is used in the same way on these systems.

20.4 What of the statistics?

In the end, the test of Stata should be on whether it enables users to analyse their survey data effectively and completely. In considering the statistical aspects, differentiate between the simple analyses, of the type described in Chapters 6 to 9, and the more complex analyses considered in later chapters.

The new graphics are impressive and these are illustrated in Chapters 6 and 8. Stata's system for tables is reasonably complete, in that we could produce any table we needed. But it lacks a system for pivoting and manipulating tables that is in Excel for example. And there are no facilities for formatting tables for a report that parallels their system for making a graph presentable.

This limitation on tables is linked to our view that Stata is awkward, compared to other packages, in the way it deals with value labels for categorical data. The value labels are reasonably easy to attach, but not so simple to manipulate, in ways that would make tables more presentable.

For more complex surveys you may require good facilities for data manipulation and organisation. Stata has these in abundance, described in Chapters 10 and 12 for example. Surveys often need a weighted analysis, to scale results from the sample to the population level. We know of no other statistics package that deals with different systems of weighting as completely as Stata, see for example Chapters 12 and 13.

Stata's special guide on surveys presents a clear case for the reasons that surveys need to have special commands to combine weights with correct measures of precision that reflect the design. The surveys guide describes all the commands and menus and they are comprehensive. This area is crucial and well handled.

For general (advanced) statistics, everyone has a favourite package. We illustrate some analyses in Chapter 15, and there are many other possible ways of processing the data. We found that Stata "grows on you". It has a wide, and ever expanding range of facilities for analysis.

20.5 Overall

Overall Stata favourably impressed us for survey data analysis. Many research groups and others who have surveys to process should find that Stata is a strong option.

However, our main reason for preparing this guide was for a Central Statistical Office, not for a research group. Initially this is for CBS and UBOS, but their needs are fairly typical of government statistical offices in many countries. For them we are still slightly undecided. This is perhaps just as well, because the decision is not ours to make.

Our hesitation centres particularly round the production of routine tables which seems to be much of what is done by government statistical services. We describe what we were able to do in Chapter 7 and Chapter 9, and found Stata was not as strong or flexible as we would like. A little of the equivalent flexibility in producing graphs, applied to the production of tables would be very useful.

Thus, the flexibility of Excel's pivot tables would be useful. A full wish list might provide something close to the CTABLES command and the tabulation wizard produced in SPSS from Version 12. The SPSS tutorial outlines their facilities. We would like interactive table production and editing, presentation table production, and then easy routines to move the resulting tables into reports.

If users need more than Stata currently provides, then what are the options? One is to ask Stata themselves, and also Stata users what might be possible in the future. A second option is to use different software for routine tables, and Stata for everything else. An obvious package for tabulation is CSPRO, which is free, and also provides excellent data capture facilities. It has a

range of exporting formats and these include export to Stata. When Stata's tabulation is not enough, it is likely to be a large survey, so CSPRO plus Stata is an attractive option. This possibility is being evaluated by UBOS.

Another possibility is to use anther statistics package, in addition to Stata. Presumably SAS and SPSS are the front-runners? This could imply either that some staff become Stata users, while others use SPSS, say. Or perhaps everyone would be able to use both. We are doubtful about the latter. If it is decided that all Stata users in a national statistical office also need to add SPSS, then they must check whether the converse applies. What advantage would SPSS users gain from adding Stata? That is a different book!